conference

proceedings

International Workshop on Wireless Traffic Measurements and Modeling

Seattle, WA, USA
June 5, 2005

Jointly sponsored by
ACM SIGMOBILE and
The USENIX Association,
in cooperation with **ACM SIGOPS**

USENIX Association

Papers presented at the Workshop on Wireless Traffic Measurements and Modeling

Jointly sponsored by USENIX and ACM SIGMOBILE,
in cooperation with ACM SIGOPS

June 5, 2005
Seattle, WA, USA

Workshop Organizers

Workshop Co-Chairs

Maria Papadopouli, *University of North Carolina at Chapel Hill*
Dina Papagiannaki, *Intel Research Cambridge*

Program Committee

Mary Baker, *HP Labs*
Suman Banerjee, *University of Wisconsin—Madison*
Andrew Campbell, *Columbia University*
Paul Castro, *IBM T.J. Watson Research Center*
Christophe Diot, *Intel Research*
Tristan Henderson, *Dartmouth College*
Felix Hernandez-Campos, *University of North Carolina at Chapel Hill*
Xiaoqiao Meng, *University of California, Los Angeles*
Maria Papadopouli, *University of North Carolina at Chapel Hill*
Konstantina Papagiannaki, *Intel Corporation*
Pablo Rodriguez, *Microsoft Research Ltd, Cambridge*
Haipeng Shen, *University of North Carolina at Chapel Hill*
Suresh Singh, *Portland State University*
Leandros Tassiulas, *University of Thessaly*
Mark Yarvis, *Intel Corporation*

The USENIX Association Staff

International Workshop on
Wireless Traffic Measurements and Modeling.
Seattle, WA, USA

Sunday, June 5, 2005

Index of Authors

Message from the Workshop Chairs

Welcome to the international workshop on Wireless Traffic Measurements and Modeling (WiTMeMo '05). WiTMeMo was created to address the need for more realistic models for traffic, mobility, and access patterns. Furthermore, it serves as a forum for scientists and engineers in academia and industry to discuss their experiences and research results about all aspects of measurements and modeling of applications, usage, access, and mobility in wireless networks.

We are delighted to have a program with a keynote speech from Professor Margaret Martonosi of Princeton University, six peer-reviewed papers, two invited talks, and an exciting panel session. In the panel, we will address the reasons why researchers have been hesitant to experimentally validate their work based on wireless measurements and discuss the challenges in performing such measurements. Our goal is to enhance and accelerate the process of sharing traces, implementations, and test suites, while strengthening the current practices in wireless experimental research. We hope that sound experimental practices will greatly impact the future design of wireless networks, as well as their management and planning.

We would like to thank Dr. Apratim Purakayastha of IBM Research and Prof. David Kotz of Dartmouth College for their support in organizing this workshop and the program committee members for their hard work.

We hope that you will enjoy the workshop. Welcome to Seattle!

Maria Papadopouli, University of North Carolina at Chapel Hill
Konstantina Papagiannaki, Intel Research Cambridge
Workshop Co-Chairs

Analysis of a Wi-Fi Hotspot Network

David P. Blinn, Tristan Henderson, David Kotz

Department of Computer Science, Dartmouth College, Hanover, NH 03755

Abstract

Wireless hotspot networks have become increasingly popular in recent years as a means of providing Internet access in public areas such as restaurants and airports. In this paper we present the first study of such a hotspot network. We examine five weeks of SNMP traces from the Verizon Wi-Fi HotSpot network in Manhattan. We find that far more cards associated to the network than logged into it. Most clients used the network infrequently and visited few APs. AP utilization was uneven and the network displayed some unusual patterns in traffic load. Some characteristics were similar to those previously observed in studies of campus WLANs.

1 Introduction

In recent years, deployment of Wireless Local Area Networks (WLANs) has boomed as demand for wireless Internet access grows and IEEE 802.11 technology matures. 802.11 WLANs can now be found in offices, homes and campuses. One increasingly-popular use for 802.11 networking equipment is to provide wireless 'hotspots', that is, providing wireless Internet access in popular public places such as airports, shops and cafés. An understanding of how these hotspot networks are used can guide network design, hotspot deployments, and the development of technologies to be used on WLANs.

In this paper we present one of the first studies of a deployed 802.11 hotspot network. We collected a network activity trace lasting approximately five weeks from the Verizon Wi-Fi HotSpot network. We analyze the network in terms of users, Access Points (APs) and traffic, and compare some of our findings with those for a college campus wireless network and a corporate wireless network.

In the next section, we review related work. In Section 3, we describe the study environment and in Section 4 we describe the tracing methodology. Section 5 presents the most interesting features of the data and compares them to results obtained in previous studies of WLAN usage. In Section 6 we formulate our conclusions.

2 Background and related work

Recent studies have characterized wireless network usage in a variety of environments. Tang and Baker studied a packet radio network composed of nearly 25,000 radios distributed across three major metropolitan areas [10].

Balachandran et al. analyzed WLAN usage over three days in a conference setting [2]. Kotz and Essien examined a college campus wireless network when it was first installed in 2001 [7]. Henderson et al. returned to the same network after it had matured in 2003/2004 [6]. Two other campus WLANs that have been studied include the University of North Carolina [4, 8] and the University of Saskatchewan [9], while Balazinska and Castro analyzed usage of a corporate WLAN [3].

While hotspots are a popular topic in both the business and research worlds, we are unaware of any other papers that examined a deployed hotspot network. Balachandran et al. examined the challenges facing hotspot networks [1], while Verhoosel et al. proposed a generic hotspot business model [11].

3 The Study Environment

Network: The Verizon Wi-Fi HotSpot network (VWHN) consists of 312 APs distributed around the island of Manhattan.[1] APs are installed in the ceilings of Verizon-owned phone booths. Each AP is a Proxim OriNOCO AP-2500 802.11b AP[2], enclosed within a weatherproof box containing the AP, a DSL modem, a power regulator, and an external antenna. APs are connected to the Internet by a 1.5 Mbps downstream and 768 Kbps upstream ADSL connection. In the weatherproof boxes, the APs have a maximum range of close to 300 feet but in practice, due to environmental interference, an AP's effective range is approximately 150 feet.

Although all APs share the same SSID, the VWHN does not support roaming between APs. When moving from one AP to another within the network, a user must reauthenticate to obtain Internet access at the new AP.

Users: The VWHN is currently provided solely as an amenity service to Verizon Online (VONL) DSL and dial-up customers. Customers of these services use their VONL username and password to log on to the network. As of December 2004, 10,511 unique VONL accounts had been used to log on to the VWHN.

Test accounts were also distributed to Verizon employees, who routinely access the network for maintenance purposes. Although 30 to 40 of these accounts exist, fewer than ten were in use during the study pe-

---

[1]A full list of available Verizon Wi-Fi HotSpots organized by region is available online at https://www33.verizon.com/wifi/login/locations/locations-remote.jsp

[2]Specifications at http://www.proxim.com/products/wifi/ap/ap2500/

riod. Service technicians routinely associate and log into the network for maintenance purposes. Their usage, however, tended to skew the distribution of data and so we eliminated their cards from the study. A company named UDN uses the network to distribute files to electronic signs installed above subway entrances. Usage for UDN users was also atypical and their data have been excluded.

Authentication, Authorization, and Accounting: To obtain Internet access at a Verizon Wi-Fi HotSpot, a user must first log into Hotwire, a proprietary hotspot management system developed within Verizon. To log in, a user first associates to the AP and opens a web browser, which is redirected to a web page requesting a username and password. Access is granted upon submitting a valid username and password. Prior to login, an associated user's Authentication, Authorization, and Accounting (AAA) state is considered *pending* at the AP. After login, it is considered *valid*. A user may also have an *unknown* AAA state before sending any packets to the AP. A user in this state is treated as a pending user because they have similar access privileges [5].

A user may log out by clicking on a logout button provided to them at login or have their session terminated after 15 minutes of inactivity. In addition, Hotwire automatically logs out users logged on for over seven hours whether or not they are still sending or receiving data.

4 Methodology

We used the Simple Network Management Protocol (SNMP) to poll APs every 5 minutes from Nov 15, 2004 to Dec 20, 2004. Polls collected information on users including MAC address, AAA State, and bytes sent and received. Once received, messages were time-stamped using the poller's clock. Traffic counts were not reset by a change in AAA state. A total of 746,397 relevant records were logged.

A 5 minute interval was used to obtain data frequently without affecting AP operations. Moreover, entries in the AP-2500's Current Subscribers table are removed after approximately 10 to 11 minutes of inactivity. A 5 minute poll interval ensures that we observe most users associating to APs. In the results that follow, we round down when calculating session lengths — if a user i is seen at times t_0, t_1, but not at t_2, we assume that their session began at t_0 and ended at t_1.

During the study period, 282 of the 312 polled APs responded. The remaining 30 APs failed to respond because of technical difficulties.

There are four holes in the data caused by crashes in the data collection process: Nov. 17 to Nov. 19 (41 hours), Nov. 24 to Nov. 29 (118 hours), Dec. 4 to Dec. 5 (43 hours), and Dec. 5 to Dec. 6 (18 hours). In the following results, per-day and per- hour statistics exclude days

and hours for which only partial data is available. To build the most complete picture of the network possible, however, data for these incomplete time periods were taken into account when calculating statistics for the entire trace period. When considering quantities summed over the period of the trace, note that these numbers would be higher if the data were complete.

Users were not informed that the study was being performed. To protect privacy, individual users were not tracked, even though this may have been possible through tracking VONL accounts. To further protect privacy, to be consistent with prior similar studies, and because a VONL account does not necessarily equate with a distinct user, MAC addresses were treated as corresponding to individuals.

4.1 Definitions

AAA State: The Authentication, Authorization, and Accounting state of a card at a given AP. A card may have a *valid*, *pending*, or *unknown* state. A card has an unknown state before sending any packets to the AP [5]. Hereafter, we use the term pending to describe both the pending and unknown states because cards with these states have similar access privileges and we treated them as the same.

Card: A wireless NIC, identified by MAC address.

Valid Card: A card in a valid AAA state during a given time period at a given AP. If no period is specified, the period of the entire trace is implied. Valid cards have unrestricted access to the Internet at the AP where they are valid.

Pending Card: A card in a pending AAA state during a given time period at a given AP. Pending cards have Internet access limited to certain VWHN-related websites. A valid card is not guaranteed to be seen as pending even though it must have been pending at some point prior to login. Note that the set of pending cards is not disjoint from the set of valid cards.

Session: A session begins with the appearance of a card at an AP in a given AAA state (valid or pending), and ends when the card is either no longer at the AP or when the card changes AAA state.

Active AP: An active AP is an AP to which one or more cards are associated (regardless of the cards' AAA state) during a given time period.

Valid AP: An AP at which one or more associated cards was seen with a valid AAA state during a given time period.

Pending Traffic: Traffic generated by pending sessions.

Valid Traffic: Traffic generated by valid sessions.

Inbound: Traffic sent by the AP to the card.

**Outbound:** Traffic sent by the card to the AP.

5 Results

Over the 36-day trace period (which includes 22 complete days of data), we gathered 746,397 SNMP records. We saw 26,925 total cards, of which 1,682 were valid at one point in the trace. We summarize our results in a manner that facilitates comparing the VWHN with WLANs studied in other environments. In addition, we investigate usage characteristics of the VWHN that differ from previously studied networks.

5.1 Users

For a WLAN such as Verizon's, understanding the user is critical to building and maintaining a successful network.

Card Activity: Patterns in the number of valid cards for each day of the study strongly mirror the number of pending cards on the network for each day of the study (Figure 1). Some users have multiple sessions in a day, and so we observe approximately twice as many sessions as cards.

Figure 1: Cards and sessions per day. The cards and sessions for a day appear just to the right of its tic mark. Blank spaces represent holes in the data. Sundays are labeled. The x-axis is on a logscale.
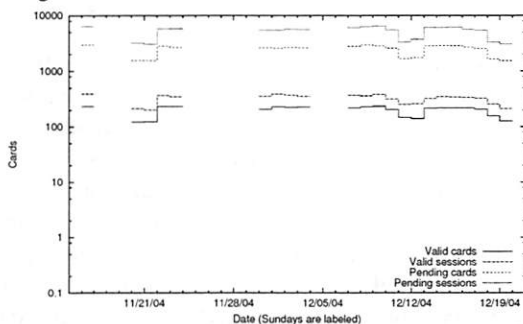

A puzzling question is raised by the small number of valid cards (1,682) in comparison to total cards (26,925) seen during the trace. Why did so many cards associate to Verizon APs but not log in (and attain a valid AAA state)? Perhaps some users are simply curious and select the VWHN SSID when they see it is an available network, or perhaps some clients' wireless networking management utilities chose to automatically associate to the network.

A median of 13% of the valid card population and 10% of the pending card population appear on any given day. A much larger portion of the user population appears daily on college [7,6] and corporate campus WLANs [3]. It appears that the VWHN is made up of many of what Balazinska and Castro term "locations visited occasionally" rather than "primary places of work" [3].

More cards are seen during the work-week than during weekends with the weekly trend for pending cards closely resembling that for valid cards (although Figure 2

shows both valid and pending cards on the same plot to save space, both valid and pending cards follow similar trends).

Figure 2: Active and pending cards per day of the week. The curve shows the mean and the bars show the standard deviation.
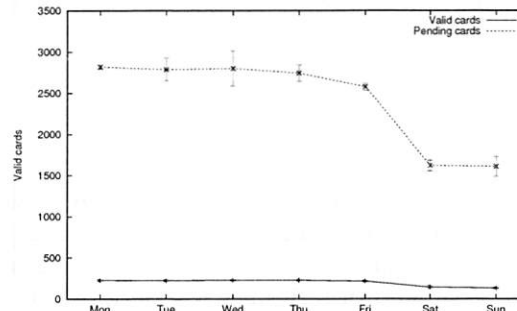
As with other wireless networks studied, Verizon's network displays a strong diurnal usage pattern (Figures 3–4). This is true for both valid and pending cards, though pending cards show greater variation in number during the busiest hours of the day. The higher numbers for pending cards during the morning commuting hours might reflect devices automatically associating as people go to work but before they begin to use the network. The number of pending cards on the network late at night is still much larger than the number of valid cards. This makes it seem unlikely that the large number of pending cards is a result of curious users. It is hard to imagine hundreds of curious users attempting to log onto an unfamiliar network late at night and in the early morning.

Figure 3: Active valid cards per hour. The curve shows the mean and the bars show the standard deviation.
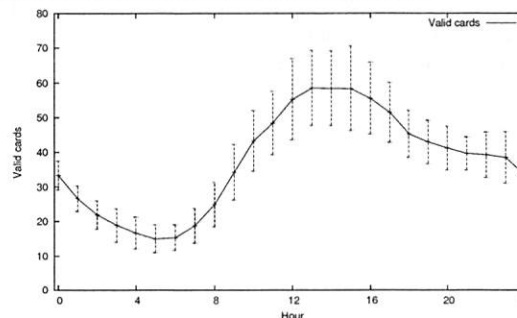
Mobility: A benefit of wireless networking is that it can enable mobility; users are not tied to a particular location by network cabling. But the opportunity for mobility does not necessarily mean that users will move around. Balazinska and Castro [3] define a user's *home location* as the AP at which a user spends more than 50% of his or her total time on the network. Adopting this definition, 95.72% of valid users had a home location, and 98.34% of pending users had a home location.

Figure 4: Active pending cards per hour. The curve shows the mean and the bars show the standard deviation.
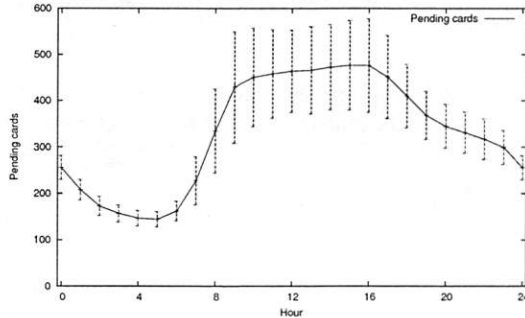
A Wilcoxon Mann-Whitney test on the distributions of time spent at the most visited AP across valid and pending cards is significant at the 1% level: more pending cards spend most of their time at a single AP than do valid cards.

23.66% of valid users and 26.93% of pending users visited more than one AP. Of these users that visited more than one AP, 81.91% of valid users and 93.84% of pending users had home locations.

In terms of home locations, the mobility of users of Verizon's WLAN more resembles that of users of a college campus WLAN [6] than that of users of a corporate WLAN [3]. APs in the Verizon network, however, are more geographically isolated from the rest of the APs in the network than APs in a campus WLAN. A card at one AP has to travel a long distance to reach another. This distance might be a cause of the high percentage of cards with home locations.

Sessions: The elbows in the distributions of valid and pending sessions (Figures 5–6) reflect the usage drops seen on weekends (Figure 1).

Figure 5: Valid sessions per day, distribution across days. Maximum: 390. Median: 346.
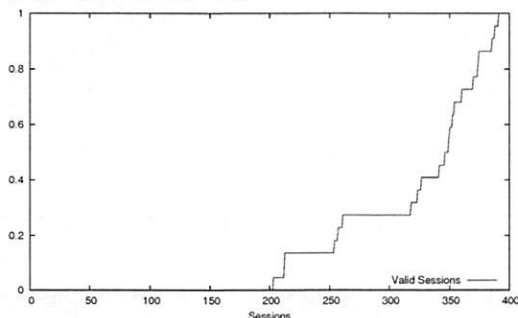
Valid sessions tend to be longer than pending sessions (Figure 7), with 45.74% of valid sessions and 12.09% of pending sessions lasting more than one hour. A log-log CCDF of the valid session durations (Figure 9(a)) indicates that session durations appear to fit a power law or Pareto distribution. The knee in the valid session distri-

Figure 6: Pending sessions per day, distribution across days. Maximum: 6468. Median: 5596.
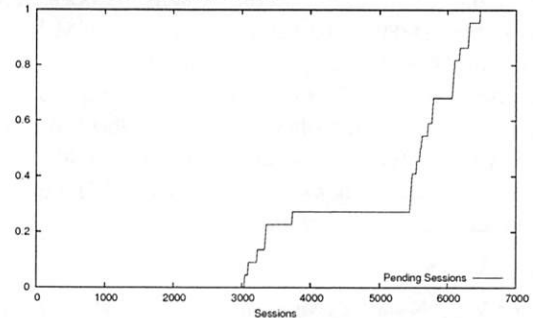
Figure 7: Session durations in hours, distribution across sessions. Maximums: 336 hours (valid), 334 hours (pending). Medians: 49 minutes (valid), 5.6 minutes (pending).
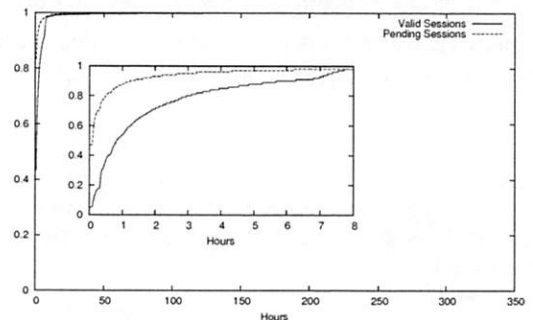
bution is caused by the fact that users are automatically logged out after seven hours (a user might appear to have a session longer than seven hours by quickly logging back in before the next SNMP poll). Considering only those sessions that last longer than seven hours, maximum likelihood estimation finds that they fit a Pareto distribution with a shape parameter $k = 1.42$ (Figure 9(b)). This is remarkably close to the session duration distribution observed on a campus WLAN [8], where a biPareto distribution is found to fit, with the long tail having a shape parameter of 1.37. We do not attempt to fit a bi-Pareto distribution to our data, as it is inaccurate at lower session durations due to the five-minute SNMP poll period, which means that short sessions are omitted from our dataset. We also find that pending session durations fit a Pareto distribution (data not shown here).

5.2 Access Points

We had 282 APs respond to SNMP polls. We now look in detail at the AP statistics.

Activity: Examining AP activity over the course of the trace, some APs see many cards while others see relatively few (figure not shown).

In testing for linear correlation (Figure 9), the proportion of variation in valid cards that is explained by the linear regression of valid cards on pending cards (r^2) is only 0.391. In other words, a device's association with Verizon's WLAN poorly correlates with the likelihood

Figure 8: Log-log CCDF (Complementary Cumulative Distribution Function) of valid session durations.
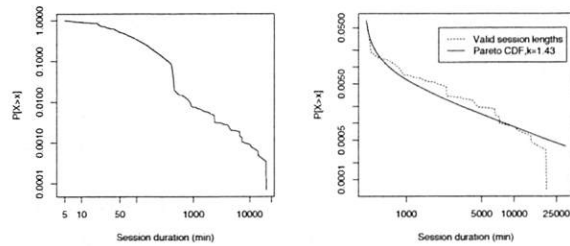
(a) CCDF of all valid session durations. The linear trend shows that the data appears to fit a power law. The knee indicates the 7 hour automatic logout.

(b) CCDF of session durations longer than 7 hours. The solid line shows a fitted Pareto distribution.

Figure 9: Scatterplot of pending cards at an AP and valid cards at an AP.
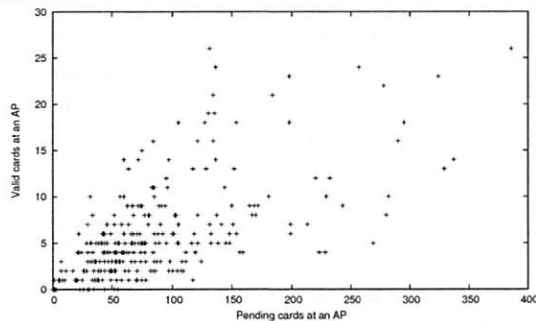
of that device actually using the WLAN. Perhaps this reflects an uneven distribution of VONL customers around the city. Or it might be that an AP's surroundings play a role in determining whether or not someone able to take advantage of the network will do so. Further investigation of the data shows that the greatest number of pending cards was seen at APs in the Midtown area, a mostly business district, while valid cards were heaviest at APs in the Upper West Side, a residential area.

Busiest periods: The hotspot APs were not particularly busy, even during peak usage periods. The greatest number of simultaneous valid sessions ever hosted by an AP was 7, whereas the most cards ever simultaneously associated to an AP was 24. The most valid cards seen by an AP during a day was 10, and the most pending cards ever seen by an AP during a day was 106. On the Dartmouth campus, in contrast, the maximum simultaneous users on one AP is 89, and the maximum cards seen on an AP in a single day is 405.

Traffic: Most APs see little traffic, but several see significant amounts (Figure 10). This pattern is similar to the traffic pattern across APs on a college campus [7, 6]

with APs handling traffic more unevenly than on a corporate WLAN [3].

Figure 10: Average daily traffic (GB), distribution across APs (CDF truncated at 1GB). Maximums: 1.56 GB (valid), 36.5 MB (pending); Medians: 4.6 MB (valid), 0.5 MB (pending).
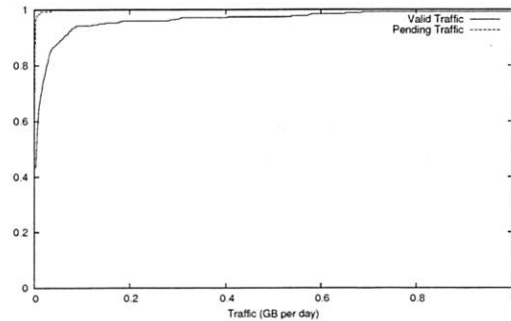
5.3 Traffic

Over the course of the trace, the network handled 281 GB of total traffic, of which 196 GB (69.9%) was inbound and 85 GB (30.1%) was outbound.

Pending Traffic: Pending traffic was mostly inbound (83.23%) although there are high outbound loads on some days (Figure 11). Pending traffic accounted for only 2.07% of total traffic. But this small percentage still totaled a median of 0.29 GB each day, which could become expensive for a hotspot provider who is paying for upstream bandwidth that is being consumed by non-customers (i.e., pending cards). Hotwire access logs show that HTTP requests from automated processes (e.g., Windows Update) being redirected to the Hotwire login page generated much of the pending traffic.

Figure 11: Daily pending traffic (GB), distribution across days. Maximums: (outbound) 0.27, (inbound) 0.53, (total) 0.80; Medians (outbound) 0.03, (inbound) 0.26, (total) 0.29.
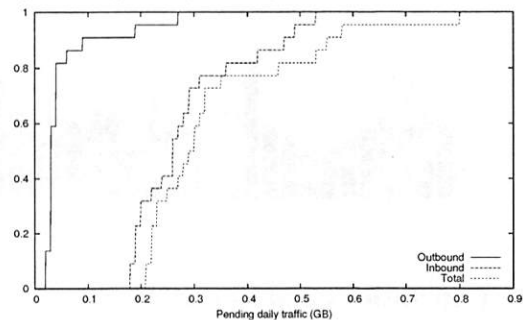
Valid Traffic: Valid traffic accounted for the majority of traffic, with 275.42 GB of valid traffic seen during the course of the trace period. Traffic per day varied moderately during days of the trace (Figure 12). The busiest 5% of valid cards accounted for 85.52% of total traffic and 95.08% of outbound traffic. Even on its busiest day

(25.50 GB), the network did not approach the average traffic loads observed on a college campus network (400 GB) [6]. Considering traffic per user, however, the average daily traffic per valid card (62.4 MB) approached that of the Dartmouth network (71.2 MB). This is interesting considering that hotspot users are limited by the capacity of the DSL connections.

Figure 12: Daily valid traffic (GB), distribution across days. Maximums: (outbound) 10.15, (inbound) 15.49, (total) 25.64; Medians (outbound) 2.60, (inbound) 7.06, (total) 9.66.
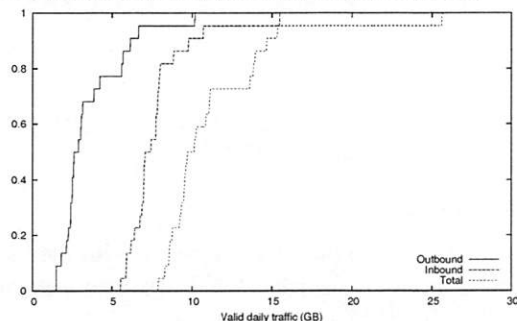
Examining valid traffic by hour, there are two peaks during the day: one in the early afternoon and one in the late evening (Figure 13). This pattern does not echo the strong diurnal pattern for valid cards shown in Figure 3. Though the midday peak corresponds with that in Figure 3, the high volume of traffic near midnight (particularly the spikes at 11 PM and 2AM) are striking. The spike at 10 AM is also odd, and was caused by an outlier: one user at a single AP on a single day.

Figure 13: Average hourly valid traffic (GB) by hour.
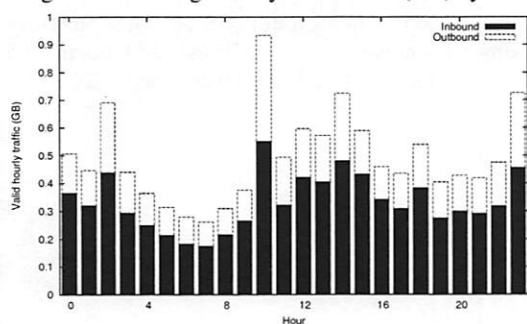
6 Conclusions and Future Work

This paper presents the first analysis of a production 802.11 hotspot network. We examine five weeks of SNMP traces from the Verizon Wi-Fi HotSpot network in Manhattan. We find that most users access the network infrequently, but daily, weekly, and hourly trends still emerge. Far more cards associate to the network than log in, and it is difficult to explain why. The vast majority of cards spend most of their time at a single AP, and few cards even visit more than one AP.

APs vary widely in their utilization. Most APs were active on any given day, but fewer saw a login. The number of cards that associated to an AP is a poor predictor of the number of users that logged in.

Most network traffic was caused by valid sessions and in particular by fewer than 5% of valid users. Traffic varied across days and exhibited unusual hourly characteristics.

We intend to look further into similarities between the hotspot network data and previously-collected campus datasets. Hotspot data is somewhat harder to obtain than campus WLAN data, and our conclusions in this study we were limited by the absence of data concerning what users were actually doing on the network along with the coarse granularity of SNMP polls. It would be useful to understand what aspects of a hotspot network can be simulated or modeled using campus WLAN data.

Acknowledgement

References

[1] A. Balachandran, G. M. Voelker, and P. Bahl. Wireless hotspots: current challenges and future directions. In *Proceedings of WMASH 2003*, pages 1–9, Sept. 2003.

[2] A. Balachandran, G. M. Voelker, P. Bahl, and P. V. Rangan. Characterizing user behavior and network performance in a public wireless LAN. In *Proceedings of the 2002 ACM SIGMETRICS Conference*, pages 195–205, Marina Del Rey, CA, June 2002.

[3] M. Balazinska and P. Castro. Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network. In *Proceedings of MobiSys 2003*, pages 303–316, San Francisco, CA, May 2003.

[4] F. Chinchilla, M. Lindsey, and M. Papadopouli. Analysis of wireless information locality and association patterns in a campus. In *Proceedings of INFOCOM 2004*, pages 906–917, Hong Kong, China, Mar. 2004.

[5] D. Fong. Nomadix quality assurance test engineer, Dec. 2004. Personal communication.

[6] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings of MobiCom 2004*, pages 187–201, Philadelphia, PA, Sept. 2004.

[7] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. *Wireless Networks*, 11:115–133, 2005.

[8] M. Papadopouli, H. Shen, and M. Spanakis. Characterizing the duration and association patterns of wireless access in a campus. In *11th European Wireless Conference*, Apr. 2005.

[9] D. Schwab and R. Bunt. Characterising the use of a campus wireless network. In *Proceedings of INFOCOM 2004*, pages 862–870, Hong Kong, China, Mar. 2004.

[10] D. Tang and M. Baker. Analysis of a metropolitan-area wireless network. *Wireless Networks*, 8(2–3):107–120, Mar.-May 2002.

[11] J. Verhoosel, R. Stap, and A. Salden. A generic business model for WLAN hotspots: a roaming business case in The Netherlands. In *Proceedings of WMASH 2003*, pages 85–92, Sept. 2003.

MobiNet: A Scalable Emulation Infrastructure for Ad Hoc and Wireless Networks

Priya Mahadevan
UC San Diego
pmahadevan@cs.ucsd.edu

Adolfo Rodriguez
IBM, RTP and Duke University
razor@cs.duke.edu

David Becker
Duke University
becker@cs.duke.edu

Amin Vahdat
UC San Diego
vahdat@cs.ucsd.edu

Abstract

The current state of the art in evaluating applications and communication protocols for ad hoc wireless networks involves either simulation or small-scale live deployment. While larger-scale deployment has been performed, it is typically costly and difficult to run under controlled circumstances. Simulation allows researchers to vary system configurations such as MAC layers and routing protocols. However, it requires the duplication of application, operating system, and network behavior within the simulator. While simulation and live deployment will clearly continue to play important roles in the design and evaluation of mobile systems, we present *MobiNet*, a third point in this space. In MobiNet, the communication of unmodified applications running on stock operating systems is subject to the real-time emulation of a user-specified wireless network environment. MobiNet utilizes a cluster of *emulator* nodes to appropriately delay, drop or deliver packets in a hop by hop fashion based on MAC-layer protocols, ad hoc routing protocols, congestion, queuing, and available bandwidth in the network. MobiNet infrastructure is extensible, facilitating the development and evaluation of new MAC layers, routing protocols, mobility and traffic models. Our evaluations show that MobiNet emulation is scalable and accurate while executing real code, including video playback.

1 Introduction

Wireless mobile systems have become increasingly popular in the past few years. Of particular interest has been the proliferation of *ad-hoc* wireless networking where mobile nodes form peer relationships with one another to relay information through the network.

One key challenge in this area is evaluating these protocols and applications in a scalable and accurate manner. It is difficult and costly to deploy development software on a large number of real mobile nodes. Further, live deployment makes it difficult to obtain reproducible results. To overcome these limitations, researchers have developed simulation engines to mimic the behavior of mobile systems by modeling packet loss, queuing de-

lays and MAC-layer behavior. Application code is typically re-written to conform to the simulation environment. This approach requires increased development effort and also leads to loss in accuracy as the behavior of a unmodified application running over a real OS, network stack and hardware is lost. Finally, accurate simulation environments face significant scalability limitations, often topping out at a few tens of mobile wireless hosts.

MobiNet is an emulation environment designed to overcome some of the accuracy and scalability challenges in mobile evaluation. The goal of our work is to allow users to evaluate the behavior of their wireless systems under a range of conditions in a controlled, reproducible environment. System aspects that we would like to allow users to control include MAC layers, routing protocols, mobility patterns and traffic models. MobiNet supports flexible deployment of the above models, thereby allowing researchers to study and improve the performance of wireless applications and protocols.

To support the above types of experiments, we designed MobiNet to emulate a target mobile network on a scalable LAN cluster with gigabit interconnect, enabling researchers to deploy unmodified IP-based software and subject it to faults, varying network conditions, different routing protocols, and MAC layer implementations. Edge nodes running user-specified OS and application software are configured at the IP-layer to route packets through one or more MobiNet *core* nodes that cooperate to subject the traffic to the bandwidth, interference patterns, congestion, and loss profile of the target network topology.

We must address several key challenges to successfully emulate large-scale multi-hop wireless networks. Behavior of MAC layer (*e.g.* various flavors of 802.11) significantly impacts the performance of wireless networks. Node mobility plays an important role in wireless environments. Ad hoc routing protocols are critical for relaying packets. Therefore our emulation supports: i) various MAC layers ii) routing protocols iii) node movement

patterns. Each of the above layers must be deployed in a modular manner, allowing users flexibility and control over their experiments. Further, we would like to structure our emulation in a scalable, accurate and extensible manner. Our scalability tests show that a single MobiNet core can forward up to 89,000 packets per second. Using just one MobiNet core and 2 physical edge nodes, we have been able to emulate a 200-node topology, forwarding application packets in real time. Along with scalability, MobiNet also provides good accuracy. We validated our MAC and routing protocols against other simulators and found that our results compared favorably with those obtained from ns2. We also present results from running unmodified binaries (video playback) that demonstrate the power and flexibility of our system.

The remainder of this paper is organized as follows: Section 2 describes the details of the MobiNet framework. We briefly describe MobiNet's accuracy and scalability in section 3. Section 4 describes our experiences in deploying real unmodified applications over MobiNet. We discuss related work in section 5 and present our conclusions in section 6.

2 The MobiNet Framework
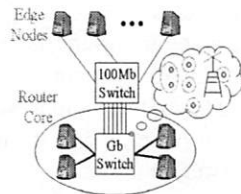
Figure 1: MobiNet Architecture

We borrowed some basic design principles from the publicly available ModelNet [10], an emulation environment for wired and static networks. However, MobiNet required a complete reimplementation of the system given the inherent differences between wired and wireless networks. The MobiNet architecture is composed of *edge nodes* and *core nodes* as shown in Figure 1. Edge nodes in MobiNet can run arbitrary architectures and operating systems and could even be a combination of different devices such as laptops, PDAs, etc. Our current experiments have been performed on edge nodes running linux. They run native IP stacks and function as they would in real environments with the exception that they are configured to route IP traffic through MobiNet cores. MobiNet core nodes run a modified version of FreeBSD to emulate topology-specific and hop-by-hop network characteristics.

Target applications run on edge nodes as they would in a real setting. However, to decrease the number of client
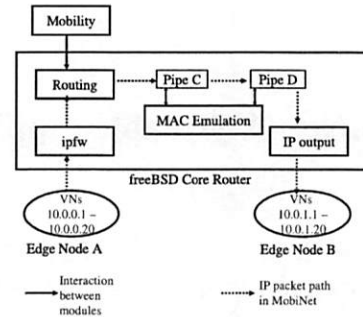
Figure 2: MobiNet Modules

(edge) machines required for large-scale evaluations, our architecture allows for *Virtual Edge Nodes* (VNs). VNs enable the multiplexing of multiple application instances on a single client machine, each with its own unique IP address. Since MobiNet clients use internal IP addresses (10.*), the number of clients that can be multiplexed onto an edge node is not limited by IP address space limitations, but rather by the amount of computational resources (*e.g.* threads, memory) that the target application uses. All VNs are configured to route their traffic through one of the cores. The MobiNet cores emulate wireless network behavior at multiple layers while eventually routing packets to the edge node hosting the destination VN.

Emulation at the core involves capturing node movement patterns, dynamic routing, and MAC layer effects such as collisions and capture. To this end, we have a mobility module, routing module and MAC layer module in the MobiNet core. The physical layer also plays an important role in wireless networks, hence we have support for free space propagation model and two-ray ground reflection model[1]. By dividing mobile emulation behavior under functional lines, MobiNet's modules are more easily developed and replaced. This allows experiments to use different combinations of modules, leading to a more flexible and powerful emulation framework. Figure 2 depicts the interactions between the different modules in MobiNet. The mobility model is implemented as a user level application that downloads new node movement files into MobiNet core's kernel at user specified time intervals. The routing module uses this information to find new routes when existing routes become stale. Once a packet enters the system, it is handed up by the *ipfw* module in the FreeBSD kernel to the MobiNet module. The routing module within MobiNet is now responsible for finding a path in order to send this packet to its destination. The path is basically a list of nodes through which the packet has to traverse before reaching its destination. Once the path has been obtained, the MAC-layer module emulates the packet according to the specified attributes of each *pipe* in the path. Pipes in MobiNet correspond to the trans-

mission capacity of their associated nodes. The packet traverses through every intermediate node's pipe, thereby being subjected to queuing delays and congestion at every node. Once the packet successfully reaches the last hop in the path, the packet is sent to the virtual node hosting the packet's destination. Thus, transmitting a packet from source A to destination B via nodes C, D, and E will involve sending the packet through pipes A, C, D and E before finally relaying it to destination B. Each pipe maintains a drop-tail queue for storing packets that need to be transmitted from the corresponding node. All attributes of pipes such as bandwidth, queue size and loss rate are user-configurable and can be downloaded into the core's kernel using the *sysctl* function call in FreeBSD.

MobiNet emulation is a three step operation: topology creation, assignment of VNs and pipes to hosts and cores respectively, and application execution. A user creates a desired topology, MobiNet distributes pipes associated with each node in the topology across the cores to distribute emulation load, assigns VNs in our emulated topology to edge nodes, and configures and executes the applications in the MobiNet emulation framework. We now describe each of MobiNet's modules in more detail.

2.1 Mobility

The mobility module is a user-level application that generates various node positions and neighbor lists consisting of nodes within a node's transmission range. This information is downloaded in real time into the kernel of the MobiNet cores at regular user-specified intervals. Alternatively, we could calculate these positions and neighbor lists in real time within the core's kernel. Doing so, however, would cause significant overhead since floating point operations would be required in the kernel.

One interesting parameter in MobiNet's emulation is that of the interval used to refresh node positions within the core's kernel. If the interval is too high, valuable kernel processing is wasted in reading new node coordinates for values that have changed little. If it is too low, it leads to inaccurate results. MobiNet attempts to bridge the gap between kernel performance and accuracy by choosing an interval value that provides good performance and accurate results under a wide variety of emulations. We found that setting the node position refresh rates to 0.5 seconds provides good results for our test scenarios, with node velocities up to 20 m/s. We stress that the refresh interval is user configurable and node coordinates can be downloaded into the kernel at a much lower granularity.

Our current mobility application supports the random waypoint mobility model described in [1], though MobiNet can use arbitrary movement models. In our applications, users specify the topology size, the duration of the experiment, the maximum speed of nodes, the movement *pause time*, and the interval of the desired output.

The mobility application creates time-indexed movement files that include the current positions of each node and the neighbor lists for each node. These movements files can be read by the MobiNet core during the execution of the experiment.

2.2 MAC Layer Emulation

Our modular emulation approach is amenable to a wide range of models for the MAC layer. We implemented our MAC layer based on IEEE's 802.11 standard specification for RTS-CTS-Data-ACK in MobiNet. The details of our implementation are described in [5]. The physical layer plays an important role in the performance and energy consumption of mobile and wireless systems. The free space model and the two-ray model predict the received power as a deterministic function of distance [1]. Our physical link model supports free space propagation and two-ray ground reflection model [1]. Power level at which packets are received determines if one or both packets are dropped due to noise or if one packet is captured by the other.

2.3 Dynamic Routing

As with all other MobiNet modules, the routing layer is implemented as a pluggable module in the FreeBSD kernel. The MobiNet core makes a call to this routing module to retrieve paths for the packets that it receives. We have implemented the Dynamic Source Route (DSR) [3] protocol in the MobiNet core. While we chose DSR in our current implementation, DSR can be replaced with any other ad-hoc routing protocol such as AODV [8], DSDV [9], or TORA [7]. Our generic design and the fact that each component in MobiNet is pluggable and not dependent on other components enable us to implement a broad range of routing modules in the kernel with relative ease. Detailed implementation is described in [5].

3 Evaluation

In this section, we briefly describe our experiences using MobiNet for evaluating ad hoc wireless applications. Our evaluation focused on testing MobiNet for scalability as well as accuracy.

We have written and tested a simple application in native TCP/IP and in the ns2 network simulator to enable comparisons between MobiNet emulation and ns2 simulation. The application establishes simple constant bit rate (CBR) streams between senders and receivers using UDP. Each sender sends data to exactly one receiver. Our CBR communications consists of 64-byte packets sent from each node (sender) at the rate of 4 packets per second. While it is impossible to guarantee that both versions function identically, the simplicity of our test application leads to it exhibiting very similar behavior in both environments. Using this application, we have ex-

ecuted a number of experiments to evaluate the performance, scalability, and accuracy of the different modules in MobiNet. The goal of our accuracy and routing overhead tests were to reproduce the experiments described in [1].

In all of our experiments, MobiNet edge nodes consisted of Pentium 4 2.0 GHz PCs with 512 MB memory running linux version 2.4.2. We use a single Pentium 3 dual processor with 2 GB memory supporting FreeBSD version 4.5 as our MobiNet core. Our experiments on ns2 were conducted on a machine similar to our edge nodes. MobiNet provides various packet statistics that enable us to determine the number of packets sent, packets dropped due to MAC collision, and other useful metrics. Likewise, we make use of ns2 trace files to extract these metrics.

With our mobility application, we simulated random waypoint mobility using various seeds and pausetime values, resulting in different movement patterns. For most of our experiments, we specified a neighbor-refresh interval of 0.5 seconds. We found that our interval of 0.5 seconds gives us comparable results with lower intervals such as 0.2 seconds and also with the continuous movement pattern that ns2 supports.

3.1 Core Performance

One of the experiments we have executed, tested the ability of the MobiNet core to process packets. The goal was to find the number of packets per second the MobiNet core router could emulate without saturation.

The setup comprised of 200 VNs that were distributed across 2 edge nodes (100 virtual IP addresses mapped to each edge node). We disabled the mobility module to decrease the overhead due to DSR. Thus DSR is invoked only once for a source-destination combination. Once a route to a particular destination has been found by the routing module, the route does not change. A sender application was associated with every VN on one edge machine, while a listener was associated with every VN on the other edge machine. Each sender application sent 64-byte UDP packets at a constant bit rate to a specific listener, thereby accounting for 100 flows from the sender edge machine to the lister edge machine. Each source sent packets in exactly 1 hop to exactly one destination which was also the node's sole neighbor. Thus, there were no packet collisions. We also set the DIFS and SIFS values in the 802.11 specifications to zero as the goal was to gauge the maximum number of packets that could be sent through a single MobiNet core. MobiNet core runs with a clock resolution of 10Khz, meaning that we are able to accurately emulate each packet hop to within 0.1 ms accuracy. Even for end-to-end path lengths of 10 hops, packet transmission delays are accurate to within 1 ms, sufficient for our target wireless scenarios, especially when considering end-to-end transmission, propagation,

and queuing delays. This accuracy holds up to and including the peak emulation rate because MobiNet's emulation runs at the kernel's highest priority level. We mea-

CPU utilization at core	Pkts/sec forwarded for 1 hop	Pkts/sec forwarded for 3 hops	Pkts/sec forwarded for 5 hops
50%	43.5K	25K	16K
70%	63.5K	38K	23K
90%	78K	47K	30K
100%	89K	50K	35K

Table 1: Forwarding capacity at the Core

sured throughput in terms of packets per second and CPU utilization at the core for different packet sending rates. We ran similar tests but with different topologies, so that each packet from the sender must traverse 3 hops and 5 hops respectively before reaching the destination. Again, we ensured that there were no collisions and nodes just had their communication partners as their neighbors. As the number of hops increased, we found that the total number of packets that the core could forward per second decreased as it now had to perform more work per packet. We summarize our results in Table 1.

3.2 MAC layer and routing accuracy

Validating the behavior of our MAC layer implementation is difficult as no known emulation or simulation technique can accurately predict the bit error rates or radio interference under arbitrary deployment scenarios. To gain some baseline confidence in the accuracy of our 802.11 MAC model, we conduct micro-benchmarks to compare MobiNet's MAC layer performance with that of ns2 for a variety of topologies and packet transmission rates. Since the packet transmission rate is dependent upon the timing and rate of collisions, we hypothesize that if MobiNet and ns2 deliver the same packet throughput under a range of conditions, the packet collision and backoff behavior is likely to be similar. We experimented with several topologies and packet sending rates. For each of our topologies, we found that MobiNet and ns2 had similar packet delivery ratio. Our detailed results for different topologies are described in [5].

The next step was to validate routing accuracy in our emulator. We achieved this by comparing experimental results obtained from MobiNet to that from ns2 for our simple CBR communication. We used the 802.11 MAC protocol and DSR implementations available in ns2. Using our mobility model, we generated movement files that were used by ns2 and MobiNet. We varied the maximum speed and pause time in our experiments and for each of the above, we found that MobiNet's packet delivery ratio matches that of ns2. We also compared the number of control packets transmitted by our DSR im-

plementation with that of ns2 in the above experiments and again found that MobiNet compared favorably with ns2. Again the experiments and results are described in [5]. All the above tests helped validate the MAC and routing accuracy of our emulator.

3.3 Scalability

Given the accuracy of our emulation experiments, we next consider the scalability of our emulation environment. One of the main benefits of MobiNet over using a simulator such as ns2 is experiments can be run in realtime. Simulators that do not run in realtime have an advantage that however complex the experiment, it eventually completes. On the other hand emulators that run in realtime find the load too great at some stage. However, for typical experiments, MobiNet is capable of forwarding up to 89,000 packets per second and thus has a distinct advantage over ns2 with respect to time taken to complete experiments upto this capacity.

To quantify this benefit, we compare the time required to run experiments in ns2 and MobiNet as a function of numbers of CBRs. We used a 200 node topology with nodes distributed randomly in a 3000 meter by 600 meter rectangle (resulting in the same node density as our previous experiments). For MobiNet, the 200 nodes were distributed across 2 MobiNet edge nodes. The ns2 experiments were run on a single machine with the same configuration as the MobiNet edge node. We disabled node mobility in this case to reduce the overhead due to finding routes with DSR. Here, DSR only needs to find routes to destinations once (at the start of the experiment). We varied the number of CBR sources from 10 to 40, with each sender once again transmitting 64-byte packets at the rate of 4 packets per second. Each node sent a total of 1200 packets. Figure 3 shows the computation time necessary to execute the experiment for MobiNet emulation and ns2 simulation. This is the time it takes for the experiment to complete multiplied by the number of machines used in the experiment. In real time, this experiment takes 5 minutes, as it takes each CBR source 300 seconds to transmit its share of packets. As a result, MobiNet using 3 machines (2 edges and 1 core) emulates the experiment in 15 minutes. In contrast, ns2 simulation time of the experiment increases linearly with the number of CBR nodes. In the case of 40 nodes transmitting, the ns2 simulation lasted 134.5 minutes, compared to MobiNet's 15-minute emulation.

4 Deploying Real Applications

In this section we demonstrate the utility and generality of our infrastructure by deploying and evaluating real unmodified code, a video player over MobiNet. We used XAnim as our sample application. XAnim is a program that plays a wide variety of animation, audio and video
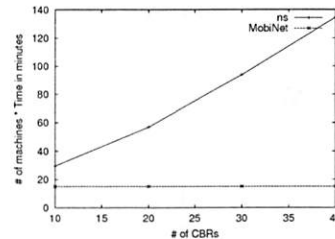
Figure 3: Scalability in MobiNet vs. ns2 as a function of time

formats on Unix X11 machines. Running the same application on ns2 would be difficult to impossible. Our goal was to study the performance of the the video player in a ad hoc wireless network as a function of node movement.

We started with a wireless topology consisting of 50 nodes moving according to the random waypoint movement model, where the maximum random speed was set to 1 m/s. The nodes in our topology were hosted on two edge machines, thus each edge node was responsible for 25 VNs. We randomly chose two VNs from our topology. XAnim was deployed over one of the VN, while the display was set to the other VN. Communication between these two nodes ran over the x11 protocol. The VN executing XAnim would send its packets to the MobiNet core, which would use DSR to find a route to the VN hosting the display. The packets were emulated according to our 802.11 implementation in the MobiNet core and then sent to the destination VN which would display the movie. Due to node movement, if existing routes went stale, DSR was used to find fresh routes to the destination VN. The video clip was replayed in a con-

Pause time (s)	1 m/s	5 m/s	20 m/s
0	14500	13708	5490
30	15596	13728	13031
60	14927	14565	13207
300	16200	16100	16086

Table 2: x11 packets exchanged between 2 VNs for various maximum speeds

tinuous fashion for 2 minutes. For lower node mobility scenarios, packet drops due to broken routes was low and we observed that the video played in an almost continuous manner. In a highly mobile environment, we found that the video clip would stall for a while during packet drops. Once routes were found, the clip would start playing again.Unlike CBR communication, in the x11 communication that takes place between the XAnim nodes, loss of vital packets due to node movement leads to the application stalling for a while. We recorded the total number of XAnim packets exchanged between the two VNs for different values of pause time and for different values of maximum speed. We averaged the results over

several runs of the experiment and present them in Table 2.

5 Related Work

Zhang and Li [12] have built an infrastructure for testing mobile ad hoc networks. However, their work does not support any routing protocol. Furthermore, their scheme does not restrict application bandwidth, making experimental results inaccurate for a range of important application characteristics. Noble and Satyanarayanan [6] use trace-based network emulation to play back measured mobile network characteristics to real applications. Our approach generalizes this technique, allowing users to generate their own mobility scenarios. Netbed [11] is a network testbed comprising real mobile nodes using real mobile hardware and software. In contrast to our work, Netbed is a real testing environment, not an emulation or simulation infrastructure. Emwin [13] and JEmu [2] are network emulators similar to MobiNet. However, they both do not have the level of scalability that we have achieved with MobiNet. There is also no support for plugging in ad hoc routing protocols. Judd and Steenkiste [4] describe an approach for wireless experimentation using a real MAC layer. While using a real MAC layer has advantages, scalability is limited as discussed above. Comparison between different MAC layers also becomes more difficult to perform.

6 Conclusions and Future Work

The overall goal of our work is to support controlled experimentation of a variety of communication patterns, routing protocols, and MAC layers for emerging ad hoc and wireless scenarios, including laptops, and PDAs. Current approaches to such experimentation include simulation and live deployment. While each clearly has its relative benefits and will continue to play an important role in mobile system design and evaluation, this paper argues for the power of modular, real-time emulation as another important point in this space.

To this end, this paper presents the design and evaluation of MobiNet, a scalable and accurate emulator for mobile, wireless and ad-hoc networks. MobiNet provides accurate mobile and wireless emulation, comparing favorably with existing network simulators while offering improved scalability. It allows researchers to rapidly experiment with a variety of MAC, routing, and communication (layers 2-4) protocols that may not be easily available in live deployments. MobiNet also supports the deployment of different mobility and traffic models. We further show the power of our emulation environment by running an unmodified video playback application communicating across an emulated large-scale

multi-hop 802.11 network using DSR on stock hardware/software.

In most of our experiments, we validated MobiNet against ns2 to increase our confidence in the accuracy of our results. We felt that this was an appropriate choice because ns2, with mobile/wireless extensions, has undergone significant development and validation and remains one of the most popular simulators available. We leave comparisons against real wireless networks for future work. A detailed study of application performance under different traffic traces is another ongoing effort.

References

[1] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, October 1998.

[2] J. Flynn, H. Tiwari, and D. O'Mahony. A Real-Time Emulation System for Ad Hoc Networks. In *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, January 2002.

[3] D. Johnson and D. Maltz. Dynamic Source Routing in ad hoc wireless networks, Mobile Computing, edited by Tomasz Imielinski and Hank Korth. pages 153–181, 1996.

[4] G. Judd and P. Steenkiste. Repeatable and Realistic Wireless Experimentation through Physical Emulation. In *Proceedings of HotNets-II*, November 2003.

[5] P. Mahadevan, A. Rodriguez, D. Becker, and A. Vahdat. Mobi-Net: A Scalable Emulation Infrastructure for Ad Hoc and Wireless Networks . In *Technical Report CS2004-0792*, July 2004.

[6] B. Noble, M. Satyananarayanan, G. Nguyen, and R. Katz. Trace-based Mobile Network Emulation. In *Proceedings of SIGCOMM*, September 1997.

[7] V. Park and M. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of INFOCOM'97*, April 1997.

[8] C. Perkins. Ad Hoc On Demand Distance Vector(AODV) routing, Internet-Draft, draft-ietf-manet-aodv-spec-00.txt. November 1997.

[9] C. Perkins and P. Bhagwat. Highly dynamic Destination Sequenced Distance-Vector(DSDV) for mobile computers. In *Proceedings of SIGCOMM 94 Conference on Communications Architecture, Protocols and Applications*, August 1994.

[10] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

[11] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, Mi Hibler, C. Barb, and A. Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

[12] Y. Zhang and W. Li. An integrated environment for testing Mobile Ad-Hoc Networks. In *Proceedings of MobiHoc*, June 2002.

[13] P. Zheng and L. Ni. EMWIN: Emulating a Mobile Wireless Network using a Wired Network. In *Proceedings of WOWMOM*, September 2002.

An Accurate Technique for Measuring the Wireless Side of Wireless Networks

Jihwang Yeo[†],   Moustafa Youssef[†],   Tristan Henderson[‡],   Ashok Agrawala[†]

[†] *Department of Computer Science, University of Maryland, College Park, MD 20742*

{jyeo, moustafa, agrawala}@cs.umd.edu

[‡] *Department of Computer Science, Dartmouth College, Hanover, NH 03755*

tristan@cs.dartmouth.edu

Abstract

Wireless monitoring (WM) is a passive approach for capturing wireless-side traffic with rich MAC/PHY layer information. WM can suffer, however, from low capture performance, i.e., high measurement loss, due to the unreliable wireless medium. In this paper, we experimentally show that WM can perform reliable and accurate measurements on wireless traffic, in actual, non-ideal channel conditions.

We demonstrate how to increase capture performance by merging traces from multiple monitoring devices. This merging enables WM to capture over 99% of the IP layer traffic and over 97% of the MAC/PHY frames in a controlled experiment. Our results indicate that WM enables reliable analysis of the collected traces, and should encourage the wireless research community to use this technique for a wide variety of research areas, such as traffic analysis, user mobility and handoff analysis, and MAC/PHY anomaly detection.

1 Introduction

With the growing popularity of IEEE 802.11-based wireless networks, it has become increasingly important to understand the characteristics of wireless 802.11 traffic and the wireless medium itself. A number of measurement studies have examined traffic characteristics in wireless networks [1, 5, 7, 8, 10]. These studies have measured the wired portion of the network, using wired network sniffers and SNMP polling. Wired network monitoring (WDM) can provide accurate traffic measurements as seen in that portion of the network. They may not, however, disclose characteristics of the wireless medium (the 802.11 MAC/PHY), as wired devices can only see the traffic that is successfully transmitted to the wired side of the AP. While SNMP-based approaches may be able to retrieve such detailed wireless MAC/PHY information through the use of a properly defined MIB (Management Information Base), most existing SNMP

MIBs for APs (MIB-I (RFC 1066), MIB-II (RFC 1213), and 802.11 MIB (IEEE Std 802.11-1999)) provide very limited visibility into MAC-level behavior. A further drawback of SNMP-based approaches is that they require an interval between SNMP polls (typically every 1–5 minutes), and it has been shown that long poll intervals may miss wireless clients that associate with APs for less than this poll interval [7].

To overcome the shortcomings of SNMP and WDM, it is necessary to sniff the wireless medium itself. We refer to this technique as wireless monitoring (WM). Like WDM, WM involves a set of devices, commonly referred to as *sniffers*, which observe network traffic, but in WM, the sniffers are equipped with wireless cards for sniffing the wireless medium. WM has recently been adopted in both wireless networking research, e.g., [9], and commercial WLAN (Wireless Local Area Network) management product development.

There are three advantages to using WM. First, WM captures detailed wireless-side traffic statistics. Second, WM provides per-frame wireless MAC/PHY information, such as 802.11 MAC headers. Third, WM does not require any interaction with the existing network, unlike WDM, where network sniffers need to be attached directly to wired switches.

The data collected by WM can be used for many purposes. Physical layer information, such as error rates, can be used to develop accurate error models for 802.11 WLANs, and for site-planning to determine the signal strengths required to achieve a certain throughput or error rate. Link-layer data, such as the characteristics of data, control and management frames, can be used to develop 802.11 simulation models, and to identify anomalies in the operation of the 802.11 MAC protocol. The overall traces themselves can also be used for emulating 802.11 networks.

WM, however, can be complicated to conduct in practice. Unreliable and varying wireless channel conditions may lead to measurement loss. The goal of this

study is to demonstrate that WM can perform reliable and accurate measurement under such non-ideal conditions. We first demonstrate how to improve the *capture performance*, that is, the amount of the actual wireless traffic captured by a particular measurement technique, by merging multiple sniffer traces. Then, through a controlled experiment, using clients with varying signal conditions, we quantify WM's capture performance in terms of IP and MAC layer statistics.

In this paper, we address all the above problems for accurate measurement technique. However, WM has another big challenge: *scalability*, i.e. that the cost and management overhead can be significant for the deployment and management of a large number of sniffers. In this work, we limit our work to the fixed number of sniffers for relatively small coverage area (e.g., WLAN in a single floor with less than 10 APs). Based on the promising results of this work, we are currently working towards addressing the scalability problem in more general WLAN environment.

The rest of the paper is organized as follows. In Section 2, we discuss previous measurement studies of 802.11 WLANs. Section 3 describes the setup and implementation of our WM system. In Section 4, we describe a controlled experiment to demonstrate the accuracy of the WM technique in capturing IP and MAC/PHY layer statistics in an actual environment. Finally, we conclude the paper in Section 5 and highlight our ongoing work.

2   Related Work

There have been several measurement studies of 802.11 WLANs. These studies typically use three types of measurement techniques: WDM (wired monitoring), SNMP and syslogs (AP system logs). WDM has been used for identifying the typical traffic mix in university WLANs [7, 8, 10], or public WLANs [1]. SNMP provides information on both traffic volume and the number of active (associated) users, and has thus been used for both traffic studies [1,8,10] and user mobility studies [2]. Syslog records detail steps of association, and have been used effectively for studying user activity patterns [5, 8].

WM techniques have been used by [4, 6] to measure packet loss and bit error rates in a non-802.11 wireless network. They used a controlled environment to measure traffic between two wireless stations. Our work differs in that it examines a production 802.11 network for MAC traffic characterization and diagnosis.

3   WM Technique

In this section we describe our methodology, in which we use multiple sniffer devices and merge multiple datasets

to improve the capture performance of the WM technique.

3.1   WM Setup

To capture wireless frames, we used three network sniffers, each comprising a PC running Linux with the 2.4.19 kernel. Each sniffer had a Prism2 chipset-based wireless network interface card; two sniffers had Demarctech DT-RWZ0-200mW-WC cards, and the third had a Linksys WPC11v3 card. To measure traffic, we used the *Ethereal* protocol analyzer (version 0.9.6) with the *libpcap* library (version 0.7). Each card was placed into 'monitor mode', which allowed the card to capture 802.11 frame information on a target channel.

The sniffers captured the first 256 bytes of each observed 802.11 frame, recording the complete view of the frame, i.e., PHY/MAC/LLC/IP/Above-IP information. PHY information, such as MAC Time and SNR (signal-to-noise ratio), can be captured using Prism2 monitor header, which is not a part of the IEEE 802.11 frame header, but is generated by the firmware of the receiving card.

3.2   Implementation of WM system

In this section, we briefly describe the WM framework, based on the techniques introduced in [11]. In that work, we demonstrated two serious drawbacks of using a single sniffer: each sniffer experiences severe loss in captured frames, and each sniffer only observes its *local view*, that is, the frames observed by one sniffer, which may differ from the AP's *global view*. Our framework aims to improve the capture performance by using multiple sniffers, placed according to SNR measurements.

3.2.1   Merging multiple sniffers

Multiple sniffers can reduce measurement loss in two ways. First, a single sniffer may not be able to observe all of the frames sent to and from a particular AP, due to radio reception and range. By using multiple sniffers, we can aggregate each sniffer's local view to create a closer approximation of the AP's global view. Second, even if a sniffer had identical radio hardware and positioning to that of an AP, it may be useful to observe the frames that the AP itself was unable to receive.

To accurately merge data from multiple sniffers, we need to be able to distinguish unique 802.11 frames for removing duplicates. We also need to prevent reordering upon merging. Reordering may occur when different sniffers observe disjoint sets of frames. For instance, if there are four frames f_{1-4} transmitted on a WLAN, and sniffer A sees f_1 and f_3, but sniffer B sees f_2 and f_4. Although each sniffer has observed their respective frames

in relative order, it is impossible to use this relative order to merge the four frames. To prevent such duplication and reordering, we need to synchronize multiple sniffers' timestamps.

Our WM framework uses 802.11 Beacon frames, which are generated by the AP, as the frame of reference for all the sniffers. Beacon frames contain their own 64-bit absolute timestamps as measured by the AP, and we can therefore uniquely identify such common beacon frames in different sniffer traces. On the timestamps of such common frames, we took one of the sniffers as a reference point and used linear regression to fit the other sniffers' timestamps to the reference sniffer.

To prevent duplication and reordering, the time synchronization error (the difference between two timestamps of different sniffers for the same frame) needs to be less than *half* the minimum gap (G_{min}) between two valid IEEE 802.11 frames. In the IEEE 802.11b protocol, the minimum gap, G_{min}, can be calculated as the 192 μs (microsecond) preamble delay plus the 10 μs SIFS (Short Inter-Frame Space) and the 10 μs minimum transmission time for a MAC frame (for the case of an Acknowledgement frame) to be a total of 212 μs. Therefore, the time synchronization error needs to be less than 106 μs. Applying linear regression for each Beacon interval ($\approx$ 100ms) on 24 hours of traces from our test setup, we measured synchronization errors on the Beacon frames from another AP. We observed a maximum error of 30 μs, which is well below the 106 μs requirement. Our setup was thus suitable for measurement using multiple sniffers.

3.2.2 Sniffer placement

We used SNR measurements to place our multiple sniffers. One sniffer was placed adjacent to the AP, to be responsible for capturing the From-AP traffic and the traffic of clients near the AP. The other sniffers were placed as close as possible to the wireless clients. Assuming that clients are uniformly distributed over the coverage area, this meant placing the sniffers so that they cover as much of the AP's coverage area as possible. Generally, if we have n sniffers to place, we split the AP coverage area into n equal areas and place the sniffers in the center of mass of these areas.

To determine the AP coverage area, we first used the SNR (obtained from Prism2 header) seen in Beacon frames from the target AP to draw the *contour lines* (as shown in Figure 1). The AP coverage area was then determined by choosing a particular SNR contour, e.g., the 15-dB contour line.

We can refine this strategy by noting that, in an environment where multiple APs are installed, the coverage area of an AP may be reduced to the *Association Area* of the AP. The Association Area of an AP is the area at
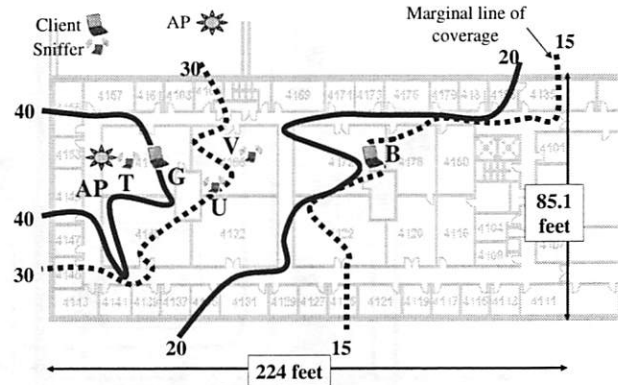
Figure 1: SNR Contour Map for controlled experiment: SNR Contour lines for 40, 30, 20 and 15 dB were obtained from SNR measurements. We placed the wireless clients at the locations with different signal conditions based on SNR measurement. Sniffers were placed at locations T, U and V.

which a client will favor this AP for association compared with other APs in the area. This behavior may be device-specific and may also vary depending on whether a client has roamed to an area or has just powered on their radio. For the purposes of sniffer placement, we assume that clients will associate with the AP with the highest SNR [9].

4 Analysis of WM Technique

In this section, we discuss the accuracy of the WM technique in capturing IP and MAC layer statistics.

For IP layer statistics, we conducted a controlled experiment, using four different measurement techniques: Application-level measurement (APP-Level), SNMP, WDM, and WM. We then analyzed the differences between WM and these other techniques. For the MAC layer statistics, we analyzed the MAC sequence numbers from the same experiment to obtain the number of missing frames in the WM results.

4.1 Controlled Experiment

4.1.1 Application level measurement

To estimate the exact measurement loss, we needed to use reliable application-generated sequence numbers. We conducted a two-way UDP packet exchange experiments using an end-to-end traffic measurement tool, *Net-Dyn* [3]. As shown in Figure 2, our NetDyn setup consisted of three different processes, *Source*, *Echo* and *Sink*. *Source* inserted a sequence number in the payload, sent the packet to *Echo*, which also added a sequence number before forwarding it to *Sink*. The *Source* and *Sink* processes ran on a wireless station, while the *Echo*

Figure 2: Controlled Experiment using NetDyn: *Source* in a wireless station sends 20,000 UDP packets to wired *Echo* machine which sends them back to *Sink* in the same wireless station.

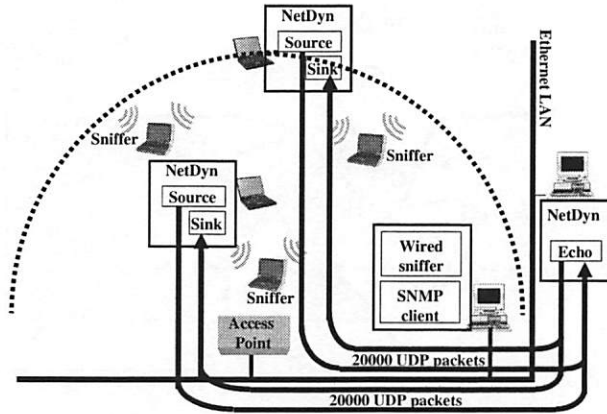process ran on a server connected to the wired LAN. Using the sequence numbers generated by the *Source* and *Echo*, we were able to determine which packets were lost in the path from the *Source* to the *Echo* and vice versa.

In our experiment, *Source* sent 20,000 1500-byte packets at 100 packets/second. We ensured that no fragmentation occurred on either side of the AP. Therefore, for each NetDyn frame on the wireless side, there was a corresponding frame on the wired side.

4.1.2 SNMP and wired monitoring

A wired sniffer running *Ethereal* was installed on the same LAN as the AP and the NetDyn *Echo* machine through a *Century Tap*, a full-duplex 10/100 Ethernet splitter. The same sniffer machine also ran a SNMP client that was configured to poll the AP for SNMP statistics every 60 seconds.

4.1.3 Experimental setup

As shown in Figure 1, we used two wireless clients at two different locations corresponding to two different signal conditions. The "Good" client G laid in an area of good AP coverage, in terms of SNR (the 40 dB-line), while the "Bad" client B laid in an area of bad AP coverage (the 15 dB-line). We also had three wireless sniffers (T, U and V) capturing the wireless traffic between *Source*, *Sink*, and the AP. Sniffer T was placed adjacent to the AP, while U and V were placed.

4.2 Application Layer Capture Accuracy

Table 1 compares the three traffic measurement techniques. Using the number of NetDyn packets (that were not lost in the paths) as the baseline (100%), we represent the traffic captured by each technique. To-AP traffic

Table 1: Comparison between APP-measurement with NetDyn, WM, WDM, and SNMP, in terms of capture percentage.

	NetDyn	WM	WDM	MIB-I	MIB-II
From	*To-AP Wireless Traffic*				
G	100	98.6	100	N/A	N/A
B	100	100.1	100	N/A	N/A
Total	**100**	**99.3**	**100**	**100.2**	**100.2**
To	*From-AP Wireless Traffic*				
G	100	99.4	103.4	N/A	N/A
B	100	102.6	103.5	N/A	N/A
Total	**100**	**100.9**	**103.5**	**102.0**	**99.9**

represents traffic from the clients to the AP, while From-AP traffic represents traffic from the AP to clients. Note that for SNMP statistics, we based our analysis on MIB-I counters as in [1] as well as the MIB-II counters (RFC 1213, RFC 2665). MIB-II provides many variables for calculating inbound/outbound error statistics more accurately than MIB-I [11].

From Table 1, we can make the following observations:

- WM has comparable performance to the other techniques for the *common* information that can be captured by other techniques, such as traffic on the wired side of the AP.

- The MIB-I and MIB-II SNMP statistics cannot reveal per-client information. The 802.11 MIBs and AP-specific MIBs may provide per-client information; we do not consider these here.

- Wired monitoring can provide accurate To-AP information about the wireless medium through the proportion of successfully-transmitted frames, as the probability of the loss on the wired medium is much lower than the probability of loss on the wireless medium. If, however, frames are fragmented on the wireless medium, we cannot obtain correct statistics on the wireless frames from the wired side.

- For the From-AP traffic, although wired monitoring can provide per-client information for the wired segment, it *overestimates* the actual traffic compared to WM. This is due to the noisy characteristics of the wireless channel, which lead to the loss of many packets on the wireless side that wired monitoring cannot capture.

- It is interesting to notice that even the SNMP statistics may differ from the true view of the wireless client. For example, in Table 1 the MIB-II total

number of successfully transmitted packets is less than the number of packets received by the NetDyn *Sink*. This can be explained by noting that there may be packets that were successfully received by the *Sink* after three retransmissions, and the corresponding MAC-level ACK was sent back. This ACK, however, was not received by the AP, and so the AP did not count it as a successful transmission.

- Due to client B's bad location, the number of successful NetDyn transmissions at client B was smaller than at client G. Using the number of successfully-received NetDyn frames as the 100% benchmark, we observe that for client B, the sniffers captured more than 100% of the NetDyn frames, e.g., 102.6% for traffic from AP to B in Table 1. In other words, for clients with bad signal strength conditions, WM can capture more frames than are successfully transmitted to/from the clients at application layer.

Finally, we note that the WM technique statistics are within **1%** of the actual application layer statistics.

4.3 MAC Layer Capture Accuracy

In this section, we examine WM's accuracy in measuring the MAC layer traffic. To count the measurement loss with only MAC frames, we exploited the IEEE 802.11 MAC sequence number. A wireless device increments the MAC sequence number whenever it sends a new Data/Management frame. We focus our analysis in this paper on the Data and Management frame types, as Control frames do not contain a sequence number. If a device retransmits a frame, then it uses the same sequence number as the original frame. Since the maximum MAC sequence number is 4095, a wireless device reuses the same sequence number every 4096 unique frames. We denote the difference in sequence numbers of two consecutive captured frames as the *gap size*. For example, if consecutive frames have sequence numbers, 4094 and 1, then there is a gap of size three ($mod_{4096}(1 - 4094)$) between the frames, and there are two missing frames between them.

Table 2 analyzes the sequence numbers for the same controlled experiments as represented in Table 1. As MAC sequence numbers are generated per device, we examined the MAC sequence numbers of the AP, client G and client B separately. Table 2 shows that as data from sniffers T, U, and V are merged, the capture performance for AP, G and B increases to 98.5%, 97.7%, and 89% respectively (with a further increase to 93% in the discussion below). The numbers in the table reflect the gap between different frames captured by the sniffers.

Table 2: Capture percentage by sequence number analysis: Merging data from sniffers T, U and V significantly increases the number of captured frames.

From	T	U	V	T+U+V	*Adjusted*
AP	97.85	96.88	96.14	98.53	98.53
G	97.48	93.65	92.92	97.72	97.72
B	61.09	88.51	88.67	88.96	**93.31**
Total	88.92	93.91	93.40	95.74	**96.97**
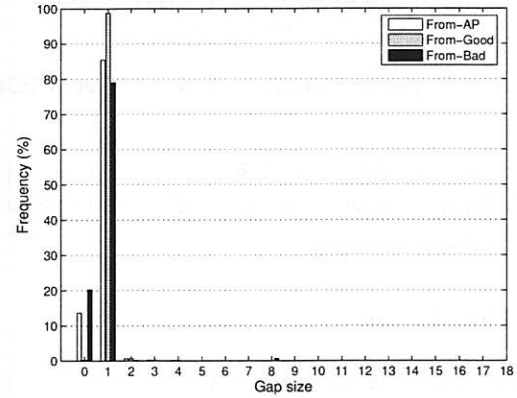
Figure 3: Distribution of sequence number gaps in From-AP, From-Good, and From-Bad MAC traffic.

Figure 3 shows histograms of the gap size between two consecutive frames for the case of using the three sniffers. Note that a gap of zero means that a frame has been retransmitted, while a gap of 1 means that there are no missing frames between these two frames. To calculate the number of missing frames, we count the gaps of greater than 1. Let $freq_i$ denote the number of occurrences of gap size i in Figure 3. Then, we can calculate the number of missing frames (N_{miss}) by $\sum_{i=2}^{i=\infty} (i - 1) \times freq_i$. The column labeled 'T+U+V' in Table 2 can be obtained by $\frac{N_{cap} \times 100}{N_{cap} + N_{miss}}$, where N_{cap} denotes the number of distinct captured frames.

Figure 4 takes a closer look at the Bad client case. The x-axis represents the received frame sequence number and the y-axis represents the gap before this frame. There was a periodic gap of 8, i.e., a measurement loss of 7 frames. By looking into the traces we found that these periodic behavior was due to the Bad client performing periodic active scanning searching for better APs. Since this process involves sending probe request frames on *different* channels [13], the sequence numbers were not captured by our sniffers, which sniffed the traffic on only one channel. If these missed probe request frames are added to the loss statistics in Table 2, the capture accuracy for the Bad client increases to 93.31% and the overall capture performance increases to 96.97% (as shown

Figure 4: Sequence number gap in From-Bad MAC traffic.

in the *Adjusted* column in Table 2). This means that the WM statistics differ by at most **3.03%** from the actual statistics.
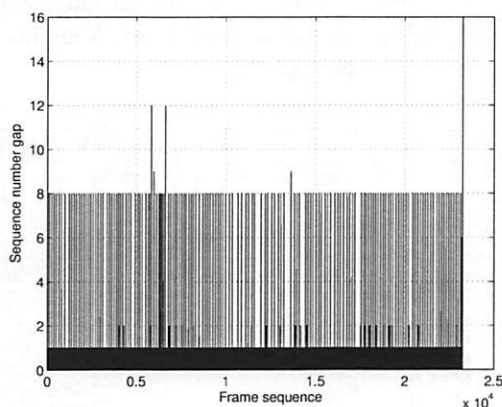
5 Conclusion

In this paper, we demonstrated that a WM technique can perform reliable and accurate measurement on the 802.11 traffic, in non-ideal channel condition. We discussed how to merge traces from multiple sniffers to increase the wireless monitoring capture performance. Results from a controlled experiment, with clients having different signal conditions, show that the WM technique captures wireless side statistics with 1% and 3% error bounds at IP and MAC layers, allowing a reliable analysis of the collected traces.

We believe that the described controlled experiment presents a worst case scenario for the WM technique. This is because our Bad client case, where the client is located on the marginal line of AP coverage, is severe and unlikely to occur in a real environment. The MAC layer capture performance of the WM technique would be much better in a WLAN environment where APs are positioned so that clients in most locations of interest can find at least one AP with good signal conditions.

Based on the results in this paper, we are currently using the WM technique to analyze the WLAN traffic in a Computer Science department environment. Some initial results can be found in [12] on traffic characterization and in [11] on anomaly detection. Besides characterizing WLAN usage patterns, we are using the traces for multiple APs to analyze user roaming patterns, co-channel interference and interactions between different APs. In such experiments, we expect that combining wired monitoring data, such as *Inter Access Point Protocol* information (IEEE Std 802.11f), with WM analysis, would give better measurement capabilities, e.g., on the roaming behavior of the mobile users and the handoff process.

We believe that our results should encourage the wireless research community to use WM techniques in many research areas, including traffic analysis, user mobility and handoff analysis, and MAC/PHY anomaly detection.

References

[1] A. Balachandran, G.M. Voelker, P. Bahl, and V. Rangan. Characterizing User Behavior and Network Performance in a Public Wireless LAN. In *Proceedings of ACM SIGMETRICS '02, Marina Del Rey, CA*, June 2002.

[2] M. Balazinska and P. Castro. Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network. In *Proceedings of MOBISYS '03, San Francisco, CA*, May 2003.

[3] S. Banerjee and A. Agrawala. Estimating Available Capacity of a Network Connection. In *Proceedings of IEEE ICON '01*, September 2001.

[4] B.J. Bennington and C.R. Bartel. Wireless Andrew: Experience building a high speed, campus-wide wireless data network. In *Proceedings of MOBICOM '97*, September 1997.

[5] F. Chinchilla, M. Lindsey, and M. Papadopouli. Analysis of Wireless Information Locality and Association Patterns in a Campus. In *Proceedings of INFOCOM '04, Hong Kong, China*, March 2004.

[6] D. Eckardt and P. Steenkiste. Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network. In *Proceedings of SIGCOMM '96*, August 1996.

[7] Tristan Henderson, David Kotz, and Ilya Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings of MOBICOM '04*, pages 187–201. ACM Press, September 2004.

[8] D. Kotz and K. Essien. Analysis of a Campus-wide Wireless Network. In *Proceedings of MOBICOM '02, Atlanta, GA*, September 2002.

[9] M. Shin, A. Mishra, and W. Arbaugh. Improving the Latency of 802.11 Hand-offs using Neighbor Graphs. In *Proceedings of MOBISYS '04, Boston, MA*, June 2004.

[10] D. Tang and M. Baker. Analysis of a Local-Area Wireless Network. In *Proceedings of MOBICOM '00, Boston, MA*, August 2000.

[11] J. Yeo, M. Youssef, and A. Agrawala. A Framework for Wireless LAN Monitoring and its Applications. In *Third ACM Workshop on Wireless Security (WiSe'04), Philadelphia, PA*, October 2004.

[12] J. Yeo, M. Youssef, and A. Agrawala. Characterizing the IEEE 802.11 Traffic: Wireless Side. In *CS-TR 4570, Dept. of Computer Science, University of Maryland*, March 2004.

[13] M. Youssef, L. Shahamatdar, and A. Agrawala. The IEEE 802.11 Active Probing Mechanism: Analysis and Enhancements. In *CS-TR-4613, Dept. of Computer Science, University of Maryland*, August 2004.

Modeling users' mobility among WiFi access points

Minkyong Kim
minkyong@cs.dartmouth.edu
Department of Computer Science
Dartmouth College

David Kotz
dfk@cs.dartmouth.edu
Department of Computer Science
Dartmouth College

Abstract

Modeling movements of users is important for simulating wireless networks, but current models often do not reflect real movements. Using real mobility traces, we can build a mobility model that reflects reality. In building a mobility model, it is important to note that while the number of handheld wireless devices is constantly increasing, laptops are still the majority in most cases. As a laptop is often disconnected from the network while a user is moving, it is not feasible to extract the exact path of the user from network messages. Thus, instead of modeling individual user's movements, we model movements in terms of the influx and outflux of users between access points (APs). We first counted the hourly visits to APs in the syslog messages recorded at APs. We found that the number of hourly visits has a periodic repetition of 24 hours. Based on this observation, we aggregated multiple days into a single day by adding the number of visits of the same hour in different days. We then clustered APs based on the different peak hour of visits. We found that this approach of clustering is effective; we ended up with four distinct clusters and a cluster of stable APs. We then computed the average arrival rate and the distribution of the daily arrivals for each cluster. Using a standard method (such as *thinning*) for generating non-homogeneous Poisson processes, synthetic traces can be generated from our model.

1 Introduction

Modeling the movements of mobile users between access points (APs) is important for simulating wireless networks. It is often not feasible to test new technologies in real wireless networks, especially not on a large scale. Simulations allow developers and researchers to try these new technologies before real-world deployment. To simulate wireless networks at the AP level, we need a model that describes movements between APs. For example, we can estimate AP load or test resource al-

location mechanisms [10] with such a movement model.

In developing a mobility model, we have three goals. First, the model should reflect real user movements. Currently available mobility models are not based on real traces and may not reflect real mobility patterns. Second, the model should be general enough to describe the movements of every device. When a user is moving, handheld devices often stay turned on, while laptops are disconnected from the network. Thus, it is not feasible to extract the physical path of laptop users by looking at network messages. Third, the model should consider the hourly variations over a day. A mobile user's movements are highly affected by the time of day, and as a result the load of APs changes over time during a day. For example, APs located at a cafeteria are visited most during lunch time. Thus, it is important to consider the hourly variations.

In this paper, we present a model of user movements between APs. From the syslog messages collected on the Dartmouth campus, we count the number of visits to each AP. Based on the observation that most APs have strong daily repetition, we aggregate the multiple days of the hourly visits into a single day. We then cluster APs based on their peak hour. We derive four clusters with different peak times and one cluster consisting of stable APs whose number of visits does not change much over 24 hours. To model a cluster, we compute hourly arrival and departure rates, and the distribution of daily arrivals. We leave the evaluation of this model as future work.

2 Clustering

In this section, we describe the traces that we used and how we discovered the period of repetition in the traces. We then describe the method of clustering APs.

2.1 Traces

We used the wireless network data collected at Dartmouth College. To observe regular student activities, we

---

chose two months—from April 1 to May 31, 2003—that did not contain a long study break. Whenever clients authenticate, associate, reassociate, roam, disassociate or deauthenticate with an AP, a syslog message is recorded. Each message contains a timestamp in seconds, the client's MAC address, the AP name, and the event type. During two months, we observed 13,888 clients associating with 533 access points.

We used a filter to convert the syslog traces into the sequence of APs that each client associates with. This filter also defines the OFF state, which represents a state of being not connected to the network. A device enters the OFF state when it is turned off or when it loses network connectivity. The latter sometimes causes devices to enter the OFF state for a short duration, lasting only a few seconds. In terms of network messages, we assume that a client becomes the OFF if it sends a disassociate or deauthenticate message. An AP also generates a deauthenticate message for a client that has not sent any message for the past thirty minutes. In this case, we consider that a client entered OFF state thirty minutes prior to the time that the deauthenticate message was generated. After conversion, our traces contain 30.1 million associations and 5.3 million OFFs.

2.2 Discovering strong period

As the first step for understanding association patterns, we counted the hourly number of users at each access point over the two months of the traces. The number of users at an AP during the ith hour is defined as $u_i = u_{i-1} - l_i + e_i$, where l_i is the number of users who left this AP during the ith hour, and e_i is the number of users who newly associated with this AP during the ith hour. As a result, we have a 1464-element vector for each AP, each element representing one hour in the 61-day period of our trace.

Instead of using a vector whose size increases linearly with the length of traces, is it possible to aggregate this information? For instance, if there is any periodic repetition, we can aggregate the values based on that period. To discover the period of repetition, we used the Discrete Fourier Transform (DFT). For each AP, we transformed the 1464-element vector from the time domain to the frequency domain using DFT. We then chose the strongest frequency (or period) signal.

The result shows that out of 533 APs, 64.5% of APs have one day as their peak period. This means that the temporal pattern repeats every 24 hours. Based on this observation, we aggregated 61 days into a single day by adding the number of visits during the same hour of different days. We then ended up with a 24-element vector for each AP. From this, we removed APs that are not actively used. We removed APs whose average hourly visits are less than three. This reduced the number of APs from 533 to 203.
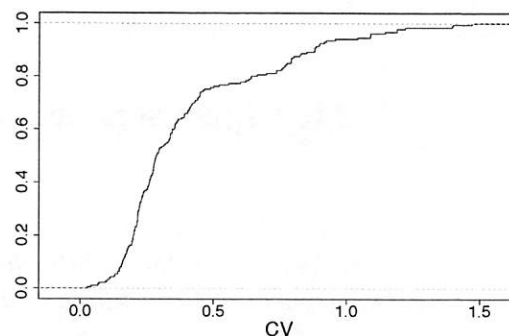

Figure 1: **Coefficient of Variation.** This figure shows the CDF of CV of standard deviation to mean of 203 APs.

2.3 Clustering APs

Our experience with the Dartmouth traces has shown that different APs have their peak number of users at different times of the day. For example, APs located at a cafeteria experience the peak during lunch time. Based on this experience, we clustered APs based on their peak hour[1].

We first want to identify the APs that are stable. Since the peak hour for these APs is not significant, these APs should not be clustered based on this value. To find the 'right' cutoff to distinguish stable APs, we plot the CDF of the coefficient of variation (CV)—ratio of standard deviation to mean—of every AP, shown in Figure 1. This figure shows a knee around the CV of 0.3. Thus, we used this value as our threshold to identify stable APs. There are 108 APs whose CV is less than 0.3. These stable APs form the *stable* cluster.

We then clustered the rest of APs based on their peak hour. As many hourly clusters have similar patterns, we merged these similar clusters and ended up with four clusters. Cluster 1 represents 11 APs with peak hours in the morning (10 AM–noon). Cluster 2 consists of 8 APs with peak hours during lunch time (noon–1 PM). Cluster 3 represents 40 APs with peak hours in the afternoon (1 PM–5 PM). Cluster 4 consists of 36 APs with peak hours in the evening (5 PM–1 AM). Note that none of APs had their peak hour in the early morning (1 AM–10 AM).

Figure 2 shows the hourly number of visits at each AP in the five clusters. The hourly visits to an AP is normalized by the total visits across the whole trace for that AP. The y-axis shows the fraction of visits that happened during each hour. In Figure 2(a), most of the APs experience a sudden increase in the number of visits at 8 AM; within two or three hours after that, this number reaches its peak. Figure 2(b) shows the APs with peak hours during lunch time. These APs have very similar patterns in hourly visits. It is interesting to note that the graph is not symmetric across 12 PM; while it increases sharply towards 12 PM, it decreases slowly after 12 PM. We expect

cluster	start time	end time	APs	diff
1	10 AM	12 PM	11	1.8%
2	12 PM	1 PM	8	0.8%
3	1 PM	5 PM	40	1.2%
4	5 PM	1 AM	36	0.5%
5	stable		108	0.3%

Table 1: **Clustered APs.** Column *Diff* shows the average difference between the hourly visits to APs and the hourly median of the corresponding cluster.

that this is due to the fact that some people have lunch late since most cafeterias on the campus serve lunch until 2 or 2:30 PM. Figure 2(c) shows the APs with peaks in the afternoon. The overall trend is having peaks in the afternoon and most of them slowly decreasing after that, while some having another smaller peak before decreasing. Figure 2(d) shows the APs with peaks in the evening. The visits of most of the APs in this cluster increase toward midnight. Figure 2(e) shows the hourly visits at the stable APs. While most of these APs experience a minimum between 5 and 6 AM as is the case with all other clusters, the number of visits does not change significantly during the rest of the day.

To show the similarity between the graphs within each cluster, we computed the average difference between the hourly median of the cluster and the hourly visits to each AP. The result is shown in Table 1. The difference result shows that Cluster 1 and 3 are noisier than the rest.

We expect that the location of APs that comprise each cluster is strongly biased. To see whether this assumption is true, we consider the types of buildings in which APs are located. We used six categories of buildings: academic, administrative, athletic, library, residential, and social [4]. Figure 3 shows the types of buildings in which APs within each cluster are located. Cluster 1, which peaks in the morning, consists mostly of academic buildings. Cluster 2, which peaks during lunch time, also consists mostly of academic buildings. This is an artifact of categorizing some of buildings that contain dining halls as academic. Cluster 3, which peaks in the afternoon, consists mostly of academic buildings and libraries. Cluster 4, which peaks in the evening, consists, not surprisingly, mostly of residential buildings. Cluster 5 of stable APs also consists mostly of residential buildings. This is because many people tend to leave their devices at home connected to the wireless network. Thus, many APs in residential areas do not experience fluctuations over the course of the day.

To examine the location of APs in more detail, we mapped the clustered APs on the campus map in Figure 4. We see that APs in Cluster 2 are in fact located around two dining areas, marked by arrows. Another interesting observation is that proximity between APs does

(a) Cluster 1

(b) Cluster 2
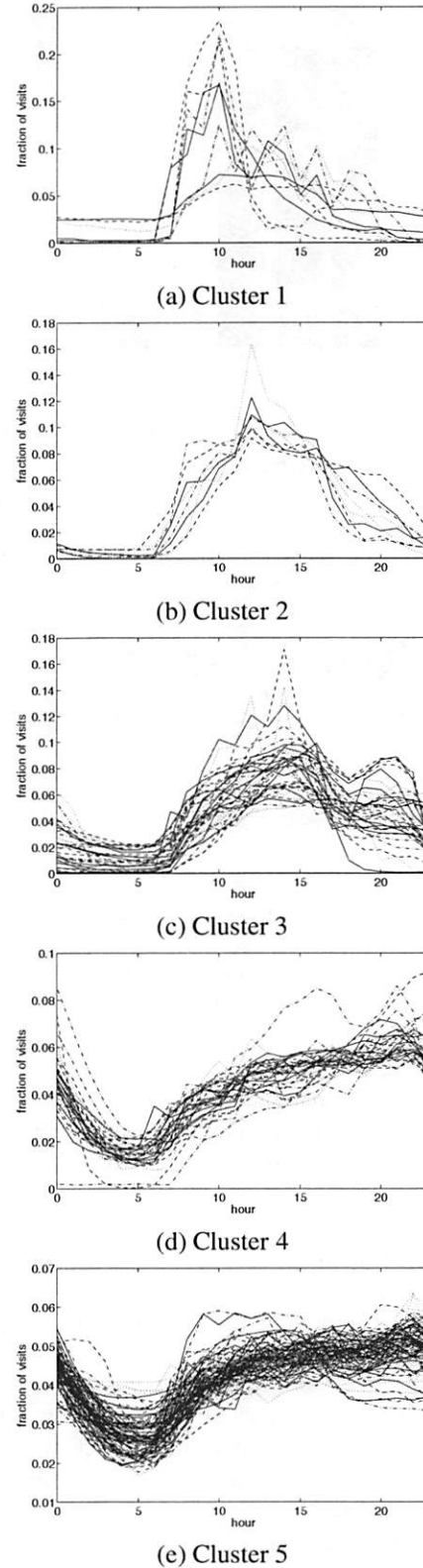

(c) Cluster 3

(d) Cluster 4

(e) Cluster 5

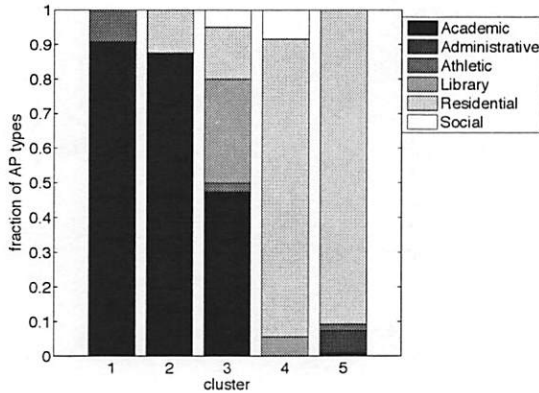Figure 2: **Normalized Hourly Visits.**

Figure 3: **Building types.** This figure shows the types APs categorized based on buildings in which they are located.
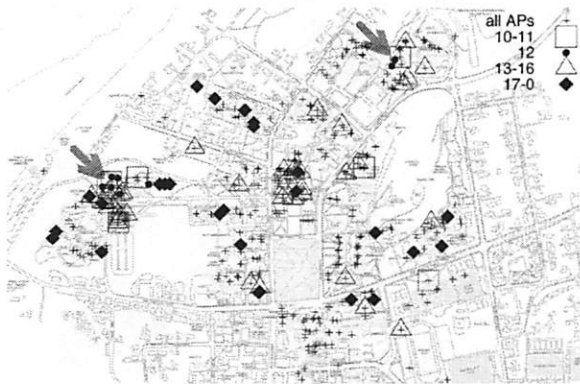

Figure 4: **APs on campus map.** This figure shows the APs and their corresponding cluster. Note that only the actively used APs are clustered. The arrows denote the two dining areas that contain the APs that peak during lunch time.

not necessarily guarantee that those APs will follow similar patterns of usage. This observation agrees with our previous study of classification of APs [7].

3 Modeling

We want to derive a mobility model that captures activity of all the wireless devices. Although the number of hand-held devices is increasing constantly, laptops still make up the majority of devices[2] in the Dartmouth wireless network. As people rarely use laptops while walking, laptops tend to be connected to the network at one location, disconnected while moving, and reconnected at another location. Due to this pattern of usage, we cannot extract the exact path of a laptop user from a source to a destination. To cope with these on-and-off devices, we developed our model of wireless network usage in terms of the arrival rate at each AP, instead of modeling the movements of individual users.

3.1 Hourly arrival and departure

To compute the arrival and departure rates, we counted the hourly number of arrivals and departures. We consider every association to each AP. For example, if the same user associates with an AP twice within an hour, both associations of that user are added to the hourly arrival value of that AP. Among the users that arrived at an AP, we considered separately those users that were previously not connected and those that were connected to the network through another AP; $A_{o,i}$ denotes the number of arrivals from the OFF state during the ith hour and $A_{a,i}$ stands for the number of arrivals from another AP during the ith hour. These two values are normalized by the total number of arrivals, $A_{total} = \sum_{i=0}^{23}(A_{o,i} + A_{a,i})$.

The average hourly departures are computed in the same way.

Figure 5 shows the average of the normalized hourly arrivals and departures for each cluster of APs. There are several interesting characteristics to note.

First, all of the clusters, except Cluster 1, have more transitions from/to another AP than from/to the OFF state. The high number of transitions from/to another AP is partly due to the *ping-pong* effect: associating repeatedly with multiple APs. When a device is within the range of multiple APs, it often changes its associated AP. Thus, changes in association do not necessarily mean that the user moved physically. The ping-pong effect is especially common where the density of APs is high.

Second, Cluster 1 has more transitions from/to OFF states than from/to another AP. We expect that this is because many faculty and students turn on their laptops during morning classes and connect to the network. Thus, these laptops make transitions from the OFF state to APs.

Third, the time lag between the arrival at and departure from APs is small. This means that the users who moved from one AP to another are not likely to stay at the new AP for more than an hour. This short duration of stay is partly due to the ping-pong effect.

Fourth, the time lag between the arrival from and departure to the OFF state is relatively big, meaning that devices tend to stay long at an AP. We expect that laptops are responsible for most of the transitions to/from the OFF state since laptop users usually close their laptops while moving. Compared to handheld devices, laptops are less affected by the ping-pong effect. Thus, the long lag is partly due to laptops being less affected by the ping-pong effect.

3.2 Daily average

Given that the arrival and departure rates that we computed in the previous section are normalized, we need the actual number of arrivals and departures to compute
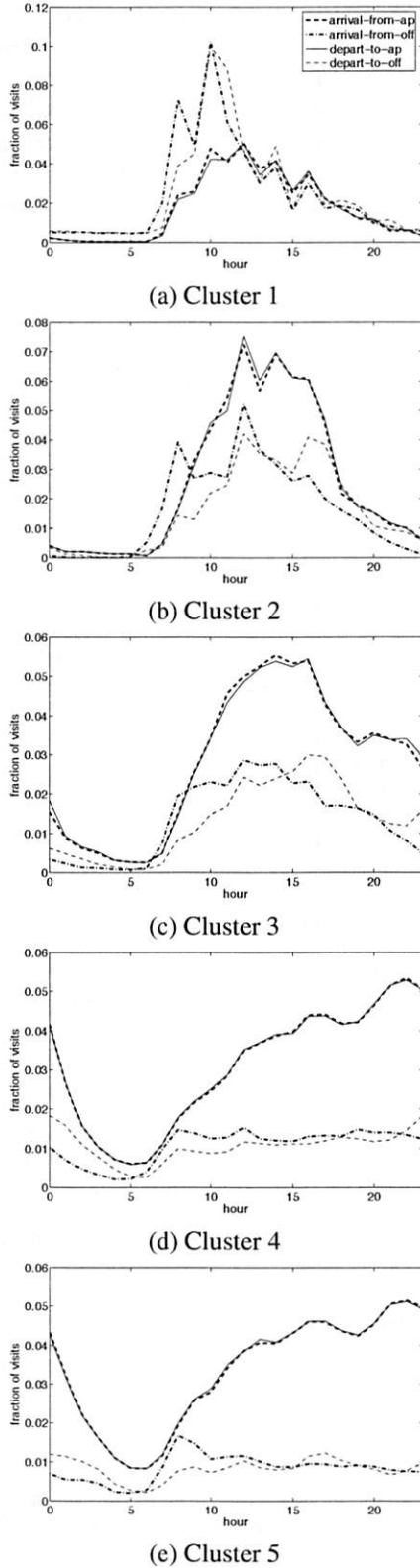
(a) Cluster 1

(b) Cluster 2

(c) Cluster 3

(d) Cluster 4

(e) Cluster 5

Figure 5: **Hourly arrival and departure.**

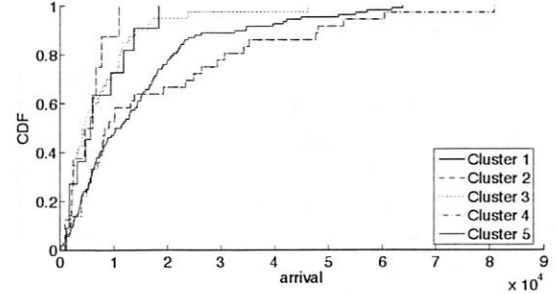| cluster	arrival	departure
1	115.177	115.171
2	84.699	84.693
3	121.363	121.334
4	298.915	298.829
5	237.434	237.333

Table 2: **Daily arrival and departure rates.**

Figure 6: **Distribution of arrivals.** This figure shows the distribution of the total number of arrivals at APs for each cluster.

the hourly value. Table 2 shows the average daily arrival and departure rates over APs within each cluster. Although the number of arrivals is larger than the number of departures, the difference is small. This implies that the number of visits at APs did not increase much during the two months of the traces.

To model the transitions between APs, the average over APs presented in Table 2 may not be enough; although APs within a cluster follow similar hourly variations, they are unlikely to have similar numbers of arrivals. Thus, we need to consider what kind of distribution the daily number of arrivals follows within each cluster. Figure 6 shows the CDF of arrivals for each cluster across all APs within that cluster.

3.3 Generating traces

Using the arrival rate and the distribution of the actual number of arrivals at each AP, we can generate synthetic traces. As our model is a process with time-varying rates (i.e., a non-homogeneous Poisson process), we can use the inversion, composition, or rejection (*thinning*) method [3] to generate synthetic traces. Since we leave the trace generation and evaluation of our model as future work, we describe *thinning* [9] only briefly.

A non-homogeneous Poisson process is determined by a rate function λ_t. Because the linear combination of Poisson processes is also another Poisson process, we can generate a time-varying Poisson process by combining multiple processes. In *thinning*, we first generate events using an exponential interarrival time with mean $1/\lambda_{max}$ where λ_{max} is the maximum rate of the time-varying process. At time t when an event is sched-

uled, the event is either accepted with the probability of λ_t/λ_{max} or canceled with the probability of $1 - \lambda_t/\lambda_{max}$.

In summary of Section 3, our model consists of the following three mobility characteristics: arrival rate following a time-varying Poisson process, departure time (or duration of stay), and the distribution of the number of arrivals at APs in each cluster. One can generate synthetic traces from our model using a standard method.

4 Related work

There have been several studies of the traces collected on the Dartmouth campus. Earlier studies [4, 8] characterize the usage of wireless networks; there was no attempt to model user mobility. Jain et al. [6] present a model of users' movements, but focus on movement only within buildings while our model describes the campus-wide movement of users.

Some more recent studies use real traces to create mobility models. Hsu et al. [5] present a Weighted Way Point model developed from a set of survey data of 268 students. Their data is limited compared to ours, which includes all wireless users on the campus. Bhattacharjee et al. [1] developed a hybrid mobility model, which favors certain directions based on probabilities computed from the observations made at only six locations on a large campus. Again, this data is limited compared to our campus-wide data. Tuduce et al. [11] developed a model from syslog traces collected on a university campus. The number of APs that a node visits is chosen from the distribution extracted from the traces. With this number, the sequence of visits to APs is chosen randomly. Thus, the model is unlikely to describe users' actual sequence of associations. Another weakness is that the movement time between APs is chosen from a uniform random distribution. The model also does not capture variations over a day.

5 Conclusion and Future Work

In this paper, we present a mobility model of the movements between APs. This model is developed using real mobility traces collected on the Dartmouth campus and reflects the real movement patterns of the wireless users on the campus. In the process of developing the model, we found that the number of visits to APs exhibits a strong daily pattern. We also found that clustering APs based on their peak time is effective; we ended up with four distinct clusters and a cluster of stable APs. We then computed the average arrival rate for each cluster and the distribution of the daily arrivals. Using a well-known method (such as *thinning*) for generating non-homogeneous Poisson processes, synthetic traces can be generated from our model.

This paper presents ongoing work. In the future, we plan to pursue several extensions. First, we would like to evaluate how closely our model describes real movements between APs by comparing the synthetic and the real traces. Second, we want to merge our model with a mobility model that describes the physical movements (or paths) of individual users. Third, we plan to explore seasonal trends such as variations between academic terms and breaks and add these trends to our model.

References

[1] BHATTACHARJEE, D., RAO, A., SHAH, C., SHAH, M., AND HELMY, A. Empirical modeling of campus-wide pedestrian mobility: Observations on the USC campus. In *Proceedings of the IEEE Vehicular Technology Conference* (September 2004).

[2] CHEESEMAN, P., AND STUTZ, J. Bayesian classification (AutoClass): Theory and results. In *Advances in Knowledge Discovery and Data Mining* (Philadelphia, PA, USA, 1996), U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds., AAAI Press/MIT Press.

[3] DEVROYE, L. *Non-uniform random variate generation.* Springer-Verlag, New York, 1986.

[4] HENDERSON, T., KOTZ, D., AND ABYZOV, I. The changing usage of a mature campus-wide wireless network. In *Proceedings of MobiCom* (Philadelphia, PA, USA, September 2004), ACM Press, pp. 187–201.

[5] HSU, W., MERCHANT, K., SHU, H., HSU, C., AND HELMY, A. Weighted waypoint mobility model and its impact on ad hoc networks - MobiCom 2004 poster abstract. *Mobile Computing and Communications Review* (Jan. 2005).

[6] JAIN, R., SHIVAPRASAD, A., LELESCU, D., AND HE, X. Towards a model of user mobility and registration patterns. MC^2R 8, 4 (Oct. 2004), 59–62. MobiHoc 2004 poster abstract.

[7] KIM, M., AND KOTZ, D. Classifying the mobility of users and the popularity of access points. In *Proceedings of the International Workshop on Location- and Context-Awareness (LoCA)* (May 2005), Lecture Notes in Computer Science, Springer-Verlag.

[8] KOTZ, D., AND ESSIEN, K. Analysis of a campus-wide wireless network. In *Proceedings of MobiCom* (September 2002), pp. 107–118.

[9] LEWIS, P. A. W., AND SHEDLER, G. S. Simulation of nonhomogeneous poisson process by thinning. *Naval Research Logistics Quarterly 26* (1979), 403–413.

[10] SONG, L., KOTZ, D., JAIN, R., AND HE, X. Evaluating location predictors with extensive Wi-Fi mobility data. In *Proceedings of INFOCOM* (March 2004), pp. 1414–1424.

[11] TUDUCE, C., AND GROSS, T. A mobility model based on WLAN traces and its validation. In *Proceedings of INFOCOM* (March 2005).

Notes

[1] We considered using AutoClass [2], a Bayesian-based clustering tool, which takes fixed-size, ordered vectors of attribute values as input. We liked the fact that it is a unsupervised classification tool (meaning that the number of classes does not need to be specified beforehand), but did not use it because it does not consider the relationship between input parameters.

[2] The earlier study of Dartmouth traces [4] shows that laptops with Windows and MacOS comprise over 76% of all the wireless devices. Smaller handheld devices including Vocera devices and Cisco VoIP phones comprise 2.5%. (The rest are unidentified.)

An Experimental Study of Multimedia Traffic Performance in Mesh Networks

Yuan Sun   Irfan Sheriff   Elizabeth M. Belding-Royer   Kevin C. Almeroth

Department of Computer Science

University of California, Santa Barbara

{suny, isheriff, ebelding, almeroth}@cs.ucsb.edu

Abstract

Performance evaluation and analysis of wireless networks is essential because testbed experiments facilitate a better understanding of network and application characteristics. This understanding of performance, in turn, results in robust protocol design. In this paper, we present an experimental study of multimedia traffic performance in mesh networks. We evaluate the performance of video and voice traffic through multi-hop wireless paths and study the capacity of the mesh network. We also investigate the impact of different traffic and network characteristics on application performance. The impact of different wireless network interface card configurations is examined, followed by our suggestions for how to improve performance. We believe our study is beneficial for both wireless network capacity planning and robust protocol design for wireless applications and services. Other researchers can also draw upon our traffic measurement experience for their own mesh testbed experiments.

1 Introduction

The growing deployment of wireless technology and infrastructure is enabling a variety of new applications. These applications require flexible and robust network support. For instance, multimedia applications, which include video streaming, VoIP and online gaming, often demand seamless real-time data delivery. These requirements, in turn, necessitate that both the application and the network be able to adapt to the highly variable nature of wireless channels. Evaluation and analysis of the performance of these applications on wireless networks therefore becomes increasingly critical so that the network and application characteristics can be better understood. Such understanding also facilitates robust protocol design for the future wireless Internet.

The majority of wireless research has been conducted using simulations which offer an efficient and flexible means to evaluate new protocols using fine-grained control. However, in simulations, MAC protocol models are often simplified, ideal wireless channels are assumed without consideration of background noise and random interference, and unrealistic traffic traces are utilized. Consequently, evaluation through simulation may not reflect the performance obtained in real networks.

As a result of the inaccuracy of simulations, many researchers have begun deploying multi-hop mesh networks for use in wireless network protocol development and testing. Testbed experiments can be challenging due to the effort required to install, configure and manage the hardware [5]. In addition, performance results are often affected by the specific configurations and protocol settings. Given the significant number of possible parameters that can affect results, finding a representative set of parameter values is non-trivial. Furthermore, the highly varying characteristics of wireless links often lead to unstable and unrepeatable results. Significant effort is necessary to enable repeatable tests and to establish adequate methods for collecting and analyzing the testbed data.

In this paper, we present our experimental study of multimedia traffic performance in mesh networks. Multimedia applications are examined because they represent a growing percentage of Internet traffic and applications. These applications demand more stringent service quality with low delay and jitter. Specifically, we perform tests consisting of video streams and voice traffic over the UCSB MeshNet testbed (http://moment.cs.ucsb.edu/meshnet). We evaluate the performance of the delivery of the multimedia data through multi-hop wireless paths and study the capacity of the mesh network. We also examine the impact of different traffic and network characteristics on application performance. Specifically, we compare the performance of bursty video traffic with constant bit rate voice traffic. We also investigate the impact of different wireless network interface card configurations. We believe our study is beneficial in both wireless network capacity planning and protocol design. We describe our analysis methodology and utilities so that other researchers can draw upon our experience for their own mesh testbed experiments.

The remainder of this paper is organized as follows. Section 2 briefly introduces the UCSB MeshNet testbed and describes the set of tools we used for our experiments. Section 3 describes the experimental setup and the evaluation metrics. Section 4 presents the experimental results and performance analysis. Finally, Section 5 discusses our observation and concludes the paper.

2   UCSB MeshNet Testbed

The UCSB MeshNet testbed is a wireless mesh network deployed on the campus of UC Santa Barbara. The network consists of 25 nodes equipped with IEEE 802.11b wireless radios. The nodes are distributed on five floors of the Engineering I building. The purpose of the testbed is to evaluate protocols and systems designed for the robust operation of multi-hop wireless networks.

The UCSB MeshNet testbed consists of two different types of nodes. Our experiments are conducted on one type of node, called *Mesh Gateways*, which are off-the-shelf Intel Celeron 2.4GHz machines running Linux version 2.4.20. The machines use wireless utilities version 16 and the hostap driver for communicating with the Netgate 2511 PCMCIA 802.11b radios. The 802.11b radios operate in ad hoc mode and connect the wireless mesh nodes. Each node is also equipped with an Ethernet interface to provide Internet access to the mesh devices and to allow out-of-band management of the mesh gateway [5].

We utilize existing tools such as iwpriv to set the pseudo BSSID and lock the cell because otherwise BSSID changes occur frequently in the tests and significantly impact the results. iptables is also used for packet filtering and route configurations. To facilitate repeatable experiments and accurate data analysis, we also developed two utilities for network monitoring and diagnosis.

Link reliability test tool: We perform link reliability tests between node pairs. The goals are to 1) measure the link quality of individual hops, and 2) identify any asymmetric links. To test reliability, the packet delivery rate in both the forward and backward direction of a link are measured. The measurements are done by sending periodic broadcast packets and recording the number of packets successfully received at each neighbor during a given period of time. Broadcast packets are used because MAC layer retransmissions do not occur for broadcast packets and thus these packets can be used to estimate the raw packet delivery rate. A link is considered symmetric if the packet delivery rate on both the forward and reverse path is above 70%. We perform each test multiple times and identify node pairs that have reliable bi-directional links. We use these node pairs for our experiments. We also verify the reliability of the links before

and after each test run to ensure that the link quality is consistent with our long term measurements.

Time synchronization tool: In our performance analysis, time synchronization between the mesh nodes is needed for delay and bandwidth calculations. The multimedia traffic cannot be utilized itself because it uses UDP as the transport layer protocol. It is thus one-way, i.e., no ACKs are provided. Therefore, the packet transmission delay needs to be measured by the destination. Further, because asymmetric links frequently occur in wireless networks, round trip latency does not provide a consistent, accurate measurement of one-way delay. Therefore, time synchronization is critical for mesh testbed experiments.

We initially applied the Network Time Protocol (NTP) [4] to eliminate the clock skew among the mesh nodes. However, our results show that the NTP synchronization precision is tens of *milliseconds*. This level of accuracy is not sufficient for our data analysis. Thus, we developed a tool to calculate the time difference of two machines by utilizing the wired management links of the mesh nodes. These links connect to the local area network in the Engineering I building. Specifically, our tool transmits consecutive 4-byte probe packets that include the timestamp of the source node. Upon reception of these probing packets, the destination node records the timestamp and echos a 4-byte packet containing the time difference between the two timestamps. At the same time, the source also sends 4-byte probe packets to measure the round trip latency. The real clock difference between the two nodes is the difference transmitted by the destination minus half of the round trip latency. We repeat the tests ten times. Our measurements indicate that, in the local area wired network, the average round trip latency and the time difference calculation have less than $10 \, \mu sec$ error.

3   Experimental Setup

In this section, we describe our experimental setup, including the network configuration and traffic characteristics of both video and voice applications. We also explain the set of experiments we performed and the evaluation metrics.

3.1   Network Topology

We utilize the reliability test tools described in Section 2 to identify the node pairs with the most reliable bi-directional connections. From the results, we select a sequence of five nodes that form a four-hop path. Two of the selected nodes are located in neighboring labs on the second floor and a third node is across the hallway. The

other two nodes are located on the third floor. We then update the routing tables of these nodes with static route entries to form paths from one to four hops.

3.2 Application Traffic

We examine the performance of multimedia traffic over the mesh network. Specifically, we use UDP video and voice streams recorded with `RTPtools` [1]. We use `rtpplay` for streaming at the source node and `rtpdump` to record the packets received at the destination. Voice traffic follows a constant bit rate (CBR) with an 80-byte voice packet transmitted every $10ms$ using G.711 codec, resulting in a data rate of $64kbps$. Video traffic, on the other hand, tends to be more bursty. Figure 1 plots a 10-second sample trace of a video source. The source transmits between two and three frames of data every second, where each frame consists of between three and seven $1KB$ packets. These packets are typically sent within a couple of milliseconds. H.261 codec is used for the video traffic and the average bit rate is $128kbps$.
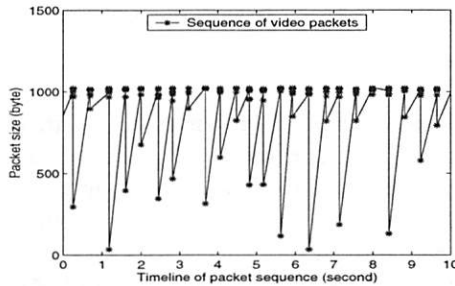

Figure 1: Sample video packet sizes transmitted by the source.

3.3 MAC Layer Configurations

In our experiments, all nodes operate in ad hoc mode on Channel 6 and use a static routing topology. The primary configuration parameters that we vary during the experiments focus on the wireless network interface cards. Specifically, we perform tests with the card operating at a *fixed* data rate ($2Mbps$) and *auto* rate (auto-rate adaptation at 1, 2, 5.5 and $11Mbps$). In the auto-rate tests, the date rate increases as the number of successfully delivered packets increases. Conversely, the transmission rate decreases when the number of packet errors increases. This mechanism is called Auto Rate Fallback (ARF) and is specified in the IEEE 802.11b standard [6]. We also investigate the impact of both the Request To Send/Clear To Send (RTS/CTS) mechanism and the maximum number of retransmissions. By default, the maximum number of retransmissions per packet is set to seven for small packets and four for large packets. RTS/CTS is recommended for large data packets[1].

---

[1]There is no specific RTS/CTS threshold value indicated in the IEEE 802.11b standard.

3.4 Experiment Scenarios and Metrics

Our tests are performed at night so that the impact of random interference (e.g., background noise, people and traffic on other wireless networks) is minimized. We also collect results during the day to examine the impact of these factors.

We conduct the following set of experiments:

1. We examine the impact of auto-rate adaptation of the wireless card by varying the data rate setting to be either fixed or auto-rate. In this scenario, the RTS/CTS is disabled and the maximum MAC retransmission number is set to seven.

2. We study the impact of RTS/CTS by comparing the performance with the RTS/CTS feature either enabled or disabled. In this scenario, the data rate is fixed at $2Mbps$ and the maximum number of retransmissions is seven.

3. We investigate the impact of the number of transmissions by varying the maximum retransmission value. In this scenario, the RTS/CTS is disabled and the data rate is fixed.

The metrics used to evaluate performance are:

1. **Packet latency**: the end-to-end packet transmission latency.

2. **Packet loss rate**: the percentage of packets that are not successfully received at the destination.

3. **Inter-flow fairness**: indicated by the variation of delay or loss among competing flows.

4. **Packet jitter**: indicated by the variation of inter-arrival latency for packets of individual flows.

4 Experiment Results

In this section, we evaluate the performance of video and audio traffic through multi-hop wireless paths and study the capacity of the mesh network. We also examine the impact of different traffic and network characteristics on the application performance. Further, we show the impact of different wireless network interface card configurations.

4.1 Capacity

Table 1 shows the number of video and voice flows that the mesh network supports as the number of hops increases. For video data, we consider less than 1% packet loss acceptable. If a more resilient coding scheme is utilized, it is possible that a higher loss rate will be tolerable. For voice data, we consider $150ms$ as the interactive voice delay threshold [3]. We tested the performance with the the NIC set to a fixed data rate ($2Mbps$) and with auto-rate adaptation.
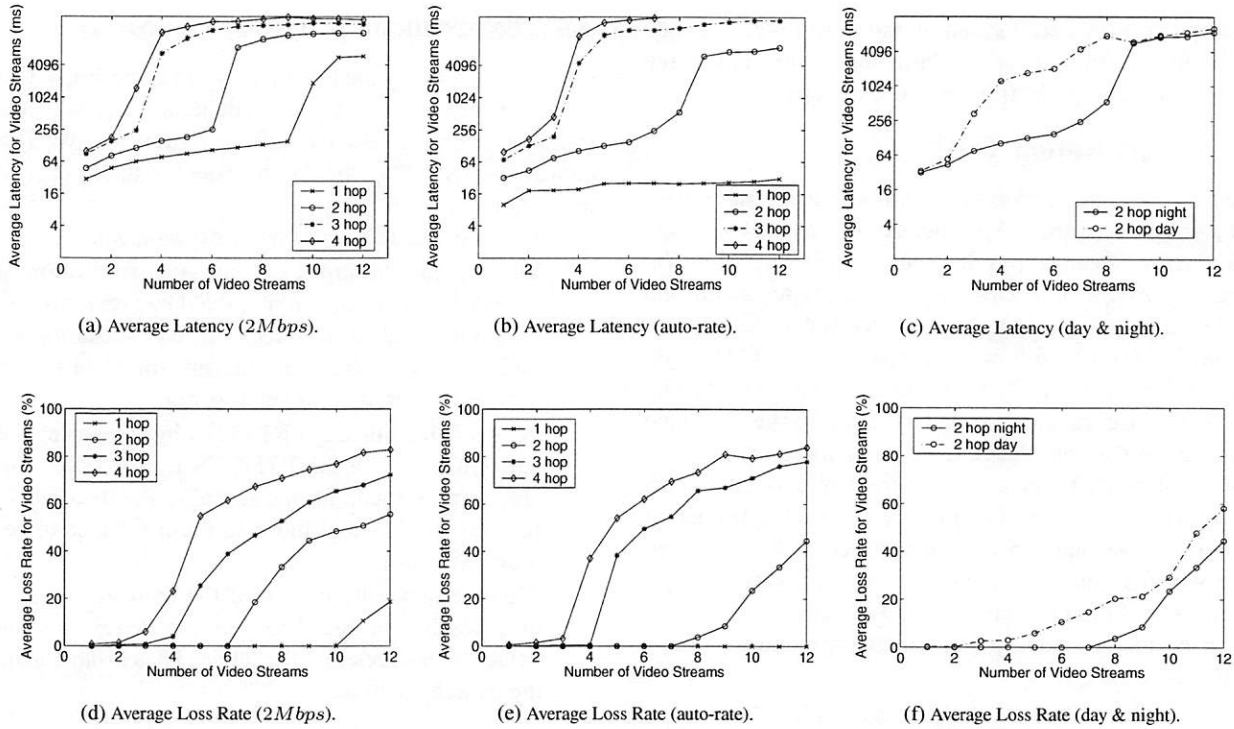
(a) Average Latency (2Mbps). (b) Average Latency (auto-rate). (c) Average Latency (day & night).

(d) Average Loss Rate (2Mbps). (e) Average Loss Rate (auto-rate). (f) Average Loss Rate (day & night).

Figure 2: Performance with increasing number of video streams.

Table 1: Number of supported, concurrent flows at acceptable quality.

Traffic	Video				Voice			
hops	1	2	3	4	1	2	3	4
Auto	30	9	3	2	11	6	3	2
Fixed (2Mbps)	10	6	3	2	11	4	3	2

Intuitively, the network should support more voice traffic flows than video traffic because voice uses a lower date rate. However, as can be seen in Table 1, this is not the case. Instead, the packet sending rate plays a more important role in determining the capacity. Specifically, voice traffic has a higher packet generation rate of 100 pkts/second, while the bursty video traffic has an average rate of about 16 pkts/second. The higher sending rate leads to network congestion, while the packet size has negligible impact on the number of supportable flows in the network. To verify the impact of packet sizes, we also performed experiments with 200 byte voice packets. This results in a bit rate of 160kbps. The results indicate that the same number of flows are supported regardless of whether the rate is 64kbps or 160kbps.

Figure 2 shows the average packet delivery latency and loss rate for video traffic with a fixed 2Mbps data rate (Figures 2(a) and (d)) and auto-rate (Figures 2(b) and (e)). We do not include the results for voice traffic because they are similar except that voice traffic in general incurs low delivery latency due to small packet sizes.

We observe that as the length of the transmission path increases, the performance degrades and the latency and loss rate increase. However, the increase is non-linear due to the increased interference from neighboring nodes. The network capacity is constrained by the number of hops. From the results, we also observe that increasing the transmission rate of the card does not necessarily increase the capacity. For instance, the number of flows supported with the auto-rate feature (with maximum rate = 11Mbps) is close to that of the fixed data rate (2Mbps) in multi-hop scenarios, especially for voice flows with a large hop count. This result occurs because of the increased contention from neighboring nodes when the path length increases. With more packet contention and subsequent packet loss, the card will automatically fall back to a lower transmission rate.

In our experiments, we notice that the auto-rate adaptation follows a slow-start-like process. All nodes operate at the lowest data rate initially. We also occasionally observe a surprisingly low video flow delivery rate for a small number of flows in the auto-rate scenario. This is because auto-rate does not always succeed in adapting to a higher transmission rate when traffic is bursty. However, once the card succeeds in adaptation, a close-to-optimal throughput of about 6Mbps can be achieved [2].

Figures 2(c) and (f) compares the performance obtained during the day and at night for video flows traversing two hops with auto-rate. Interestingly, although our test nodes operate on a different channel than the other

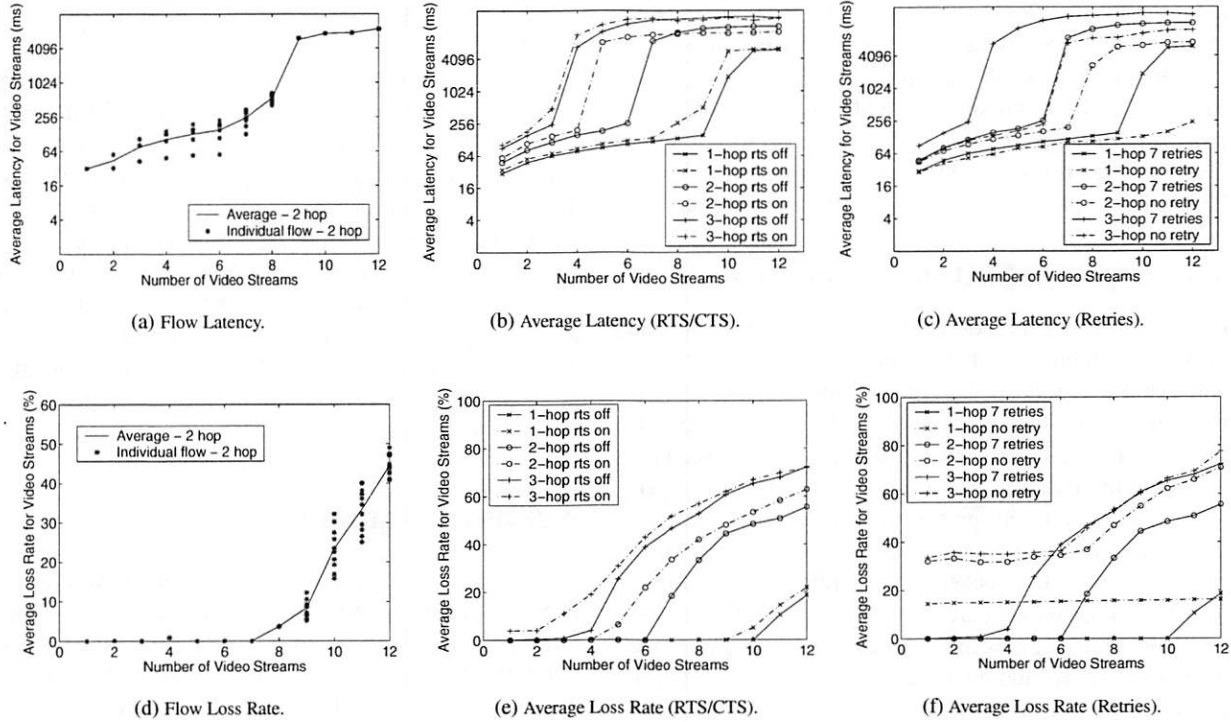| (a) Flow Latency. | (b) Average Latency (RTS/CTS). | (c) Average Latency (Retries). |
| (d) Flow Loss Rate. | (e) Average Loss Rate (RTS/CTS). | (f) Average Loss Rate (Retries). |

Figure 3: Performance with increasing number of video streams.

wireless networks in the building, we notice random interference and background noise during the day that significantly impacts the results.

4.2 Inter-flow Variation

Figures 3(a) and (d) show the fairness between competing video traffic flows when the network is operated in auto-rate mode. We notice that the latency variation among competing flows is significant when the network is not saturated. Flows started a couple of seconds after the first flows experience up to three times more latency than earlier flows. When the network is congested, the loss rate of the flows exhibits similar trends. As the path length increases, the variation becomes more significant due to the inter-flow contention between neighboring nodes. The same patterns with voice data are also observed during our experiments. These results indicate the phenomena of "channel capture" by earlier admitted flows resulting in unfairness to later flows [7].

4.3 Intra-flow Variation

Figure 4 illustrates the per packet delay for one individual flow on a 2-hop connection. The gray line indicates the delay when the network is lightly loaded with four concurrent flows. The delay variation is in the range of $5ms$ to $200ms$ with an average of $48ms$. The black line indicates the delay when the network is more heavily

loaded with eight concurrent flows. Hence there are more significant variations in the range of $6ms$ and $1200ms$ with an average of $412ms$. This indicates that with different channel conditions, traffic jitter could severely impact the received video/voice quality.
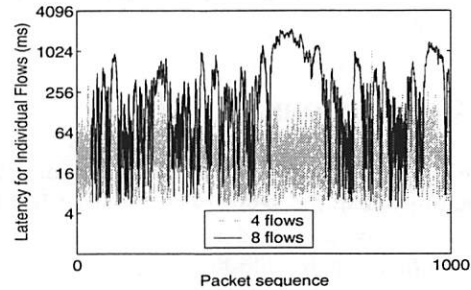

Figure 4: Intra-flow packet variation.

4.4 Impact of RTS/CTS

RTS/CTS is recommended in the IEEE 802.11 standard to eliminate the hidden terminal problem. The standard also suggests that for small packets RTS/CTS should not be utilized because of its extra overhead. For larger packets, RTS/CTS should be beneficial as a collision avoidance mechanism. Figures 3(b) and (e) show the impact of RTS/CTS by comparing the performance of video traffic with RTS/CTS enabled and disabled. The results indicate that even with large video packets, RTS/CTS does not usually offer a performance improvement in terms of

reducing latency and loss. On the contrary, it may actually limit the capacity of the network. For instance, Figures 3(b) and (e) show that in the 2-hop scenario, only four flows achieve satisfactory quality when RTS/CTS is enabled, while the network can support up to six flows when this feature is disabled. Our results suggest that RTS/CTS should not be used for multimedia traffic.

4.5 Impact of MAC Retransmissions

Figures 3(c) and (f) indicate the effect of changing the maximum number of MAC layer retransmissions on the delay and loss rate of the video traffic. A small maximum retransmission value reduces the transmission latency over each hop, as shown in Figure 3(c). Such reduction subsequently increases the capacity of the network if latency is the primary metric in consideration. The introduction of MAC retransmissions also significantly improves the packet delivery rate. As seen in the one hop scenario in Figure 3(f), the loss rate when no retransmissions are enabled is constant, indicating possible background noise and interference. When retransmission is enabled, the loss rate significantly drops. However, there is no one ideal value for the maximum number of retransmissions. When the network becomes congested, the loss rate with no retransmissions is actually no more than that with a maximum of seven retransmissions. The difference also varies with the number of hops. Hence, investigation of the relationship between the maximum number of retransmissions and the number of hops of the path would help to find an optimal value to achieve better performance.

5 Conclusions

In this paper, we have presented our experimental study of multimedia traffic delivery in the UCSB MeshNet testbed. We have evaluated the performance of video and voice traffic through multi-hop wireless paths and studied the network's capacity. We also examined the impact of different traffic and network characteristics on the performance. To summarize, we have made the following observations:

- The capacity of the network is constrained by the number of hops in the transmission path.
- The number of flows supported by the network is mostly heavily influenced by the packet sending rate, not by the data rate or packet size.
- Auto-rate adaptation does not always lead to capacity improvement when bursty traffic is present.
- Channel capture can result in unfairness among competing flows.
- Packet jitter variations can be significant in current 802.11b networks. Solutions are needed to dampen the variation for real-time traffic delivery.

- RTS/CTS does not typically help in improving performance of real-time traffic.
- Finding an optimal value for the maximum retransmission number may help improve performance.

We believe our study is beneficial for both wireless network capacity planning and protocol design. We have described our analysis methodology and utilities so that other researchers can draw upon our experience for their own mesh testbed experiments. We plan to continue our work studying experimental results obtained through our testbed. Specifically, we want to explore the techniques of reducing packet jitter in multimedia delivery and apply more advanced codec schemes and subjective evaluation methods to our traffic analysis.

Acknowledgment

References

[1] Henning Schulzrinne at Columbia University. RTP Tools (Version 1.18). http://www.cs.columbia.edu/IRT/software/rtptools/.

[2] J. Jun, P. Peddabachagari, and M. Sichitiu. Theoretical Maximum Throughput of IEEE 802.11 and its Applications. In *Proceedings of the IEEE International Symposium on Network Computing and Applications*, pages 249–257, Cambridge, MA, April 2003.

[3] C. Lin, H. Dong, U. Madhow, and A. Gersho. Supporting real-time speech on wireless ad hoc networks: inter-packet redundancy, path diversity, and multiple description coding. In *Proceedings of the ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH)*, pages 11–20, New York, NY, 2004.

[4] D. L. Mills. Internet Time Synchronization: The Network Time Protocol. In *Global States and Time in Distributed Systems*. IEEE Computer Society Press, 1994.

[5] K. Ramachandran, K. Almeroth, and E. Belding-Royer. A Novel Framework for the Management of Large-scale Wireless Network Testbeds. In *Proceedings of the 1st Workshop on Wireless Networks Measurements (WinMee)*, Trentino, Italy, April 2005.

[6] IEEE Computer Society. IEEE Standard for Wireless LAN-Medium Access Control and Physical Layer Specification, November 1999.

[7] N. Vaidya, P. Bahl, and S. Gupta. Distributed Fair Scheduling in a Wireless LAN. In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCOM)*, Boston, MA, August 2000.

A Measurement Study of Path Capacity in 802.11b based Wireless Networks

Tony Sun, Guang Yang, Ling-Jyh Chen, M.Y. Sanadidi, Mario Gerla
Department of Computer Science, UCLA
{tonysun, yangg, cclljj, medy, gerla}@cs.ucla.edu

Abstract

The rapid deployment of wireless networks in various environments necessitates the development of new end-to-end tools that monitor and measure the properties of wireless paths well. In this paper, we implement AdHoc Probe, a recently proposed path capacity estimation tool specially designed for the multi-hop ad hoc wireless environment. We present an implementation of AdHoc Probe in Linux; discuss challenges in its deployment, and solutions to the various system issues encountered. We also evaluate the behavior/effectiveness of AdHoc Probe in various testbed setups; including an interfered setting that cannot be simulated. Experiment results validate the workings of AdHoc Probe and offer insights into how the capacity of a wireless path changes in real wireless environments. Our efforts provide a basis for realistic results that can be of assistance in activities such as capacity planning, protocol design, performance analysis, and etc.

1. Introduction

As the wireless Internet grows in size and connection complexity, new applications and services requirements necessitate the development of suitable end-to-end tools that monitor and measure the properties of wireless paths well. In particular, effective evaluation and measurement of the capacities along wireless paths are of realistic interest, especially to activities such as capacity planning, protocol design, performance analysis and system deployment.

Although the path capacity estimation problem has been extensively studied in the literature, most tools work in the scenarios of wired and/or last-hop wireless networks and measure the bottleneck link capacity, e.g. [1][5][8][10]. The complexity and convergence time required for these schemes are not well suited for multi-hop ad hoc wireless networks. Moreover, the assumption of bidirectional setup of some of the above techniques has proved to yield detrimental results in ad hoc networks.

In fact, end-to-end path capacity estimation in ad hoc wireless networks is much more challenging. Wireless capacity estimation depends not only on the rate of the "narrow" link along the path (as in a wired network), but also on the topology, path layout, interference between nodes along the path and several other environmental parameters. Moreover, wireless capacity estimation must accommodate both fixed rate wireless networks, and auto rate wireless networks, where the transmission rate can be dynamically adjusted to the propagation characteristics and energy requirements. An accurate capacity estimation mechanism must understand how the capacity of a wireless path is impacted by these factors respectively.

Previous technique proposed by Li et al in [11] addressed the adhoc end-to-end path capacity by sending a brute force UDP packet stream to measure the maximum achievable throughput. This scheme produces very realistic results, but is not very practical since it heavily impacts existing traffic, and its result is affected by current on-going traffic conditions as well.

AdHoc Probe, a recently proposed adhoc path capacity estimation tool by Chen et al [1], is a simple and effective technique that aimed to simplify the adhoc path capacity estimation process. However, performance of AdHoc Probe was only studied through NS2 [19] simulations in [1]. Little understanding exists on the challenges in deploying AdHoc Probe, as well as the effectiveness and behavior of AdHoc probe in real wireless environments. Therefore, it is the purpose of this study to reveal how AdHoc Probe would perform in real wireless environments.

Motivated by the above goals, we implement AdHoc Probe in Linux and conduct in-depth evaluations in various controlled wireless testbed, to assess the efficacy of AdHoc Probe in a real network deployment. Important system issues commonly associated with such end-to-end measurement tools, such as time synchronization and clock skew, are discussed with proposed solutions. We also evaluate the effect of auto rate wire-

less radios experimentally by varying the distances between nodes as well as interfered settings/patterns that cannot be simulated via NS2. An important differentiation between auto rate radio and multi-hopping, in terms of the impact on wireless path capacities, is also among our goals.

Our experiment results show that AdHoc Probe is able to accurately measure the wireless path capacity in all cases of fixed rate networks. Moreover, AdHoc Probe is able to track the dynamic rate adaptation of an auto rate wireless link in a timely and accurate manner.

The rest of the paper is organized as follows. In section 2, we briefly describe the AdHoc Probe mechanism. An in-depth discussion of related system issues and their solutions is presented in Section 3. In section 4, we present testbed experiment results to evaluate the efficacy of AdHoc Probe in both fixed and auto rate wireless networks. Finally, section 5 concludes the paper.

2. AdHoc Probe Overview

AdHoc Probe is an end-to-end tool that measures the capacity of a multi-hop wireless path in an ad hoc network. Inspired by CapProbe [8], AdHoc Probe relies on the packet-pair technique to provide capacity estimation in wireless networks. However, while CapProbe is designed to estimate the *bottleneck link* capacity in a round-trip fashion, AdHoc Probe intends to estimate the *end-to-end path* rate based on *one-way* measurements. The end-to-end path rate is the *maximum achievable rate* over the wireless path in the absence of any competing traffic. The maximum *achievable rate* (in the absence of competing traffic) is typically lower than the nominal channel transmission rate due to a variety of reasons including multi-hopping, unique features of wireless networks (e.g. RTS/CTS mechanism), link rate adaptation techniques, etc. AdHoc Probe is able to accurately measure such achievable rate.

The basic one-way AdHoc Probe algorithm is derived from the round-trip CapProbe mechanism. Probing packet pairs of fixed size are sent back-to-back from the sender to the receiver. The sending time is stamped on every packet by the sender, the One Way Delay (OWD) is calculated at the receiver, and the path capacity (i.e. rate) estimation is performed at the receiver and communicated to the sender.

The receiver measures the OWD of each packet as the difference between the receiving time (clocked at the receiver) and the sending time (stamped by the sender

in the packet header). The OWD sum is then computed. The "good" packet-pair samples (i.e. the packet pairs encountering no cross traffic) are those with minimum OWD sum (as shown in Figure 1), and the corresponding capacity is given by $C=P/T$, where P is the packet size and T is the dispersion of the packet pair.

AdHoc Probe does not implement the "convergence test" as CapProbe does in order to make the algorithm simple, fast, and timely to the highly varying characteristics of wireless networks. Instead, AdHoc Probe simply reports the capacity estimation after receiving a certain number of samples, say N. Similar to CapProbe, the accuracy of capacity estimation increases as N grows. However, a large N value is not suitable for mobile wireless networks as it will lead to high latency in estimation and may not allow us to capture the dynamic properties of the wireless network.
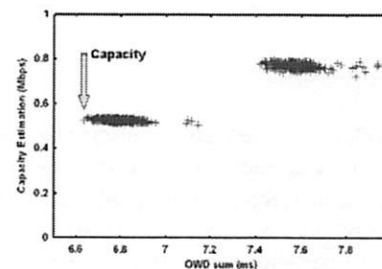
Figure 1: AdHoc Probe capacity estimate using the sample with minimum OWD sum.

Apart from the number of samples, N, the latency of the estimate also depends on the probing packets sending rate (i.e. the probing rate). For simplicity, AdHoc Probe simply sends probing packets (with the packet size of P bytes) using a CBR rate (i.e. R packet-pair/second, or equivalently $2*P*R$ bytes/second). As a result, the expected duration for one estimate is approximately N/R seconds. Clearly, the larger R is, the less time a capacity estimation process takes. However, R should be upper-bounded since a large R may disturb the ongoing foreground traffic in the network or even congest the network. As a result, the capacity estimate may become inaccurate (hard to get one good sample) or extremely slow (packets are lost due to congestion).

The probing parameters N and R need to be carefully tuned in accordance with the underlying network properties and by trading off precision for speed. This trade-off clearly depends on the application. In this paper, we simply set $N = 200$, $P= 1500$, and $R = 4$ sample pairs/second for the testbed experiments.

---

3. System Issues

In this section, we will discuss the seriousness of system issues encountered and present solutions that mitigate those problems. In subsection 3.1, we will first discuss existing Auto Rate schemes in wireless radios, allowing us to properly investigate and distinguish how Auto Rate radios and wireless multi-hopping affects wireless capacity, respectively. We will then discuss the clock screw problem in subsection 3.2. Finally, we describe the system time synchronization issue and report the implementation details of necessary "correction" in subsection 3.3.

3.1. Wireless Auto Rate

In our experiments, it is important to be aware of how existing Auto Rate schemes work, so we can properly investigate how Auto Rate radios influence wireless capacity. Auto Rate functionality is important for multi-rate wireless devices to maximize the utilization of network resources. For instance, by simply adjusting the transmission data rate, one can achieve higher data throughput with the higher data rate mode, or increase the transmission range and robustness against interference by using a lower data rate mode. Additionally, even within the same data rate mode, the overall data throughput can be improved by opportunistically taking advantage of the channel coherence time (duration for which the wireless node has better-than-average channel conditions). Finally, data rate can be changed to save energy [15].

Several auto rate schemes have been proposed to exploit the multi-rate capability. They can be categorized into two types: *Adaptive Rate* schemes (e.g. ARF [7], RBAR [4], and AARF [9]) and *Opportunistic Scheduling* schemes (e.g. OAR [16] [18] and MAD [6]).

Adaptive Rate schemes could be either sender based or receiver based. Auto Rate Fallback (ARF) [7] is the first published and implemented sender based rate adaptation algorithm. The basic idea of ARF is to start the transmission using highest rate and switch to lower rate when 1 or 2 consecutive failures occur. ARF also starts a timer upon rate dropping. When either the timer expires or 10 successfully received acknowledgements are counted, the transmission rate is upgraded to a higher rate and the timer is reset. The drawbacks of ARF are: (a) the heuristic based ARF cannot adapt effectively in a rapidly varying wireless channel, and (b) ARF data rate tends to suffer from high oscillation even when the wireless channel is not rapidly changing. In [9], Mathieu et al propose Adaptive ARF (AARF) to adapt

the threshold settings of ARF in accordance to the channel conditions. As a result, the frequent rate oscillations in ARF are mitigated.

3.2. System Time Synchronization Issue

The OWD measurement in AdHoc Probe is indeed problematic on a real testbed. Unlike the perfect time synchronization provided by the simulator, the system clocks of the two end hosts are usually not synchronized. As a result, the measured OWD will not be identical to the actual OWD. Though some software packages and service protocols (e.g. NTP [12]) have been proposed to enable time synchronization of network hosts, one can not guarantee the two end hosts are always synchronized before the estimation. Thus, a successful capacity estimation technique should not rely on any assumptions of a perfectly time-synchronized system. We now provide simple analysis on the AdHoc Probe algorithm and show that AdHoc Probe works well even when the system time is not perfectly synchronized.

Suppose δ is the constant time offset between the AdHoc Probe sender and receiver. For the i-th packet pair sample, the sending time is stamped $T_{send,i}$, and the receiving times (on the receiver) are $T_{recv1,i}$ for the first packet and $T_{recv2,i}$ for the second packet, respectively. Therefore, the measured OWD sum (S') and the actual OWD sum (S) of the i-th packet pair sample are:

$$S_i' = (T_{recv1,i} - T_{send,i}) + (T_{recv2,i} - T_{send,i}) \tag{1}$$

$$\begin{aligned} S_i &= (T_{recv1,i} - T_{send,i} - \delta) + (T_{recv2,i} - T_{send,i} - \delta) \\ &= S_i' - 2\delta \end{aligned} \tag{2}$$

Thus the difference between S_i and S_i' is a constant 2δ for all packet pairs. If $S_k = \min(S_i)$ for $i = 1...n$, $k \in [1...n]$, then $S_k' = \min(S_i')$ for $i = 1...n$ and vice versa. By filtering out those samples of non-minimum S', it is easy to identify the "good sample" that has the minimum S' and S, and the capacity estimation is computed by using the interval between this packet pair sample. Clearly, the interval is not affected by the time offset. Therefore, AdHoc Probe is able to absorb the constant time offset δ between the sender and the receiver and produce an accurate capacity estimate.

3.3. Clock Screw Issue

In additional to the time synchronization issue, the deployment of one-way AdHoc Probe may also suffer from the clock skew problem, i.e. the clock "drift" is not identical on different machines. Figure 2 shows an example of actual OWD measurements, when we send

UDP packets (4 packets per second) from one laptop to the other using 802.11b. The relative OWD (i.e. OWD_i-OWD_1 for the i-th measurement) can be skewed by 1 millisecond after only 80 packets (i.e. 20 seconds)! This is a very big error relative to the typical delay sum, in the order of tens of ms (as seen in Figure 2). As a result, when the OWD measurements are affected by an increasing (decreasing) skew, early (late) sample tend to get selected by AdHoc Probe as the "good" sample.
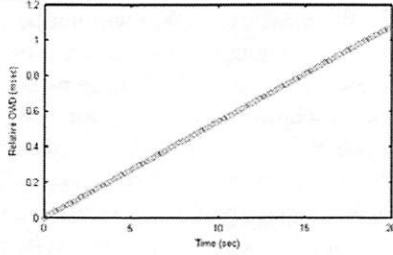
Figure 2: Illustration of clock skew problem in OWD sum measurements.

More specifically, when the clock drift is skewed, the time offset of the two machines is not constant. We use $\Delta(t)$ to denote the time offset function, where t is the system time at the receiver. The actual OWD sum of the i-th packet pair sample is then revised as

$$S_i = (T_{recv1,i} - T_{send,i} - \Delta(T_{recv1,i})) + (T_{recv2,i} - T_{send,i} - \Delta(T_{recv2,i})) \quad (3)$$
$$= S_i' - \Delta(T_{recv1,i}) - \Delta(T_{recv2,i})$$

Unlike the result in the previous subsection, $S_k' = \min(S_i')$ for $i = 1...n$ does not infer $S_k = \min(S_i)$ for $i = 1...n$. It turns out that the sample with the minimum measured OWD sum is not guaranteed to be the "good sample", which has the minimum actual OWD sum. Therefore, in order to obtain the "good sample", it is necessary to *calibrate* the measured OWD sum and remove the effect of $\Delta(t)$.

The calibration problem has been first studied in [14], in which Paxson used forward and backward path delay measurements to deal with this problem. However, this approach requires some heuristic tuning, and it is not feasible for pure one-way estimation. The linear regression algorithm is discussed in both [13] and [14]. However, it only works well if the network delays are normally distributed. [13] also formulates this problem as a linear program and solves it using standard algorithms in [3]. In this study, we employ a convex-hull based approach [17], which gives results similar to the linear programming approach, and in addition can handle clock resets and velocity adjustments as well. Moreover,

the complexity of the convex-hull based approach is $O(n)$, and is easily embedded in AdHoc Probe.

Here, we assume the time offset function $\Delta(t)$ is monotonic with either an increasing or a decreasing trend. The trend of $\Delta(t)$ can be determined by the *alg_trend* algorithm, as shown in Figure 3 with a manual threshold setting K. The smaller K is, the more sensitively the algorithm behaves. In this paper, we set $K=N/3$ for all experiments.

```
Algorithm alg_trend:
  trend = 0;
  For i = 2 to N
    If S'_i > S'_{i-1} then trend++;
    If S'_i < S'_{i-1} then trend--;
  End for
  If trend>K, Δ(t) is with increasing trend
  If trend<K, Δ(t) is with decreasing trend
```

Figure 3: Algorithm for determining the trend of offset function, $\Delta(t)$.

Once the trend of $\Delta(t)$ is found, the convex-hull calibration algorithm is applied. As shown in Figure 4, the calibration algorithm [17] will find a lower bound of skewed measurements when the trend is increasing or an upper bound when the trend is decreasing. We assume $\Delta(t)$ is a piecewise linear function; therefore, it is sufficient, but not necessary, that all good samples lie on the lower/upper bound curve.
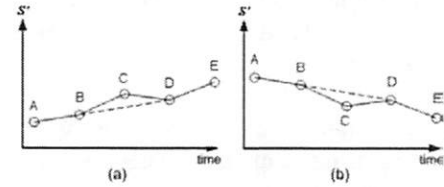
Figure 4: Illustration of convex-hull based approach. (a) the increasing trend case; (b) the decreasing trend case.

Suppose Ω_1 denotes the set of data points lying on the bound curve of OWD measurements of the first packet in the probing samples, and $\Omega_{S'}$ denotes the set of data points lying on the bound curve of S'. Ideally (i.e. each data point in Ω_1 is with the minimum OWD of the first packet, and each data point in $\Omega_{S'}$ is with the minimum OWD sum), $\Omega_{S'} \subseteq \Omega_1$. However, since Ω_1 and $\Omega_{S'}$ may also contain samples that are "not good", $\Omega_{S'} \subseteq \Omega_1$ does not always hold. In order to improve the accuracy of AdHoc Probe estimation, we identify the good samples by taking the intersection of Ω_1 and $\Omega_{S'}$, since a good sample must have both the minimum OWD of the first packet, and the minimum OWD sum. For each sample in the intersection set of Ω_1 and $\Omega_{S'}$, one capacity esti-

mation is made accordingly. AdHoc Probe then reports its end-to-end path capacity estimation result by taking the average of those estimates.

4. Testbed Experiments

Here, we perform testbed experiments to measure path capacities of wireless ad hoc networks using AdHoc Probe. Implementation issues such as time synchronization and clock skew are addressed. We experiment with AdHoc Probe in both fixed rate and auto rate actual wireless configurations in the lab. Auto rate adjustments are induced by varying the physical distances between nodes and by subjecting the 802.11b links to Bluetooth interference.

4.1. Experimental Results in Fixed Rate Wireless Networks

The testbed was first set to validate the path capacity on multi-hop fixed rate wireless networks. We placed several 802.11b laptops about 70 ~ 80 meters apart in a chain topology. The wireless rate was fixed at 2Mbps. 20 capacity estimates were collected for each path length (i.e. each number of hops). Each run included 200 packet pair samples, and 4 samples were injected every second. The experiment is conducted without cross traffic, and the average and standard deviation of the capacity estimates is presented below in Figure 5.
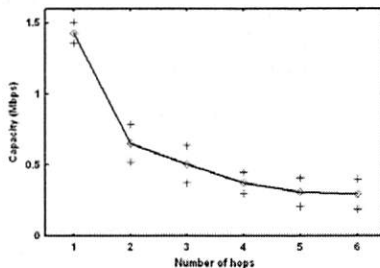
Figure 5: Experiment results of AdHoc Probe on wireless multihop testbed (transmission rate is 2Mbps on each link).

From the results, it is obvious that the effective capacity of a chain topology decreases as the hop length increases, and the estimate remains constant after the number of hops becomes larger than 4.

4.2. Experimental Results of Auto Rate Wireless Networks Triggered by Displacements

To experimentally validate the relationship between source-destination distance and path capacity, we measured the path capacity between auto rate capable nodes when the distance varies by 20 meter increments. The data transmission rate can adapt in the range {11Mbps, 5.5Mbps, 2 Mbps, 1 Mbps}. Four AdHoc Probe samples were collected every second and each run consists of 200 samples. The experiment was conducted without cross traffic. 20 capacity estimates were collected, and their average and standard deviation are presented below in Figure 6.
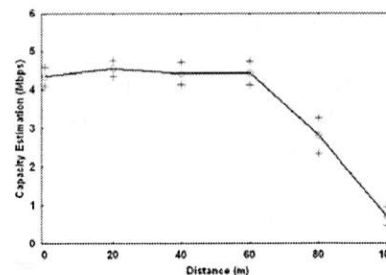
Figure 6: Experiment results of 802.11b one hop connection (auto rate) with different distance between two hosts.

From the results, the estimated capacity remains basically unchanged when the source-destination pair is within 0-60 meters (the average effective one-way capacity is approximately 4.4Mbps, which corresponding to the 11Mbps modem rate when the various O/H components are factored out). When the distance between the source and the destination node increases beyond 60 meters, we observed a decrease in the measured capacity. In particularly, when the distance between the source-destination reaches 80 meters, AdHoc Probe measures an average effective capacity of ~3Mbps, corresponding to 5.5 Mbps modem rate. When the distance between source-destination reaches 100 meters, we measured an average effective capacity of ~1Mbps, which again, corresponds to 1Mbps modem rate[1]. The experimental results thus confirm the relationship between source-destination distance and path capacity.

4.3. Experimental Results with Bluetooth Interference

Rate adaptation can be triggered not only by a change in distance but also by wireless interference. In fact, interference has the same effect as reducing the signal to noise ratio as distance does.

---

[1] The effective capacity of a one-hop 802.11b link can be found at http://www.uninett.no/wlan/throughput.html

To investigate the influence of wireless interference on effective capacity of a wireless link, we set up an experiment with a single hop 802.11b path interfered by Bluetooth. Figure 7 illustrates the testbed configuration. Two 802.11b laptops (i.e. AdHoc Probe sender and receiver) are placed 10 meters apart, and two Bluetooth laptops (the interference generators) communicate with each other creating interference to the 802.11 receiver. The Bluetooth pair is placed at a varying distance from the 802.11 receiver (from 0 to 9m). The Bluetooth source sends a CBR traffic to the Bluetooth receiver at 240kbps (1500 bytes/packet; 20 packets/second). Since Bluetooth and 802.11b use the same radio frequency band (i.e. 2.4GHz), they interfere with each other, and the link quality of the 802.11b connection degrades. As a result, the 802.11b sender adjusts its rate using ARF attempting to adapt to the changing channel conditions.
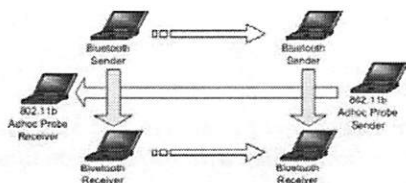
Figure 7: Auto Rate 802.11b Testbed with Bluetooth interference.

For each data point， 20 AdHoc Probe tests were made, each test consisting of 200 packet pair samples. Probing rate is 5 packet-pairs per second. The average and standard deviation of the capacity estimates are presented below in Figure 8.
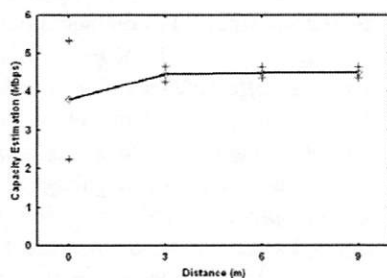
Figure 8: Experiment results of 802.11b with auto rate and with Bluetooth interference from varying distance.

From the results, the average capacity estimate is consistently in the 4Mbps range, which is what we expect for a single hop 11Mbps channel. The estimate is very sharp for Bluetooth beyond 3m. For zero distance, the estimate oscillates as the Auto Rate controller tries to keep up with the changes. It is remarkable that the average estimate at zero Bluetooth distance is quite close to the actual rate.

5. Conclusion

In this paper, we have implement and conduct an in-depth evaluation on the effectiveness of AdHoc Probe, a newly proposed end-to-end path capacity estimation tool designed for multi-hop ad hoc wireless networks. Important system issues associated with the mechanism, such as time synchronization and clock skew, were discuss in detail, and solutions were presented. Effects of the auto rate wireless radio on wireless path capacity are also studied and evaluated experimentally by varying the distances between nodes as well as the interference patterns. We have also investigated the respective effects of auto rate radio and multi-hopping on wireless path capacities. Our experiment results have shown that AdHoc Probe is a useful and practical tool that can indeed be deployed in real wireless networks. It is able to accurately measure the wireless path capacity in all cases of fixed rate networks, and track the rate adaptation of an auto rate wireless link timely and correctly.

Acknowledgement

References

[1]   Chen, L.-J., Sun, T., Yang, G., Sanadidi, M. Gerla, M., *"AdHoc Probe: Path Capacity Probing in Ad Hoc Networks"* UCLA Computer Science Technical Report TR050005.

[2]   Dovrolis, C., Ramanathan, P., and Moore, D., *"What do packet dispersion techniques measure?"* IEEE Infocom'01.

[3]   Dyer, M. E., "Linear Algorithms for Two- and Three- variable Linear Programs," SIAM Journal on Computing, 13 (1983), 31-45.

[4]   Holland, G., Vaidya, N., and Bahl, P., "A rate-adaptive MAC protocol for multi-hop wireless networks," MobiCom 2001.

[5]   Jacobson, V., *"Pathchar: A tool to infer characteristics of Internet paths"*, ftp://ftp.ee.lbl.gov/pathchar

[6]   Ji, Z., Yang, Y., Zhou, J., Takai, M., Bagrodia, R., "Exploiting Medium Access Diversity in Rate Adaptive Wireless LANs," MobiCom 2004.

[7]   Kamerman, A. and Monteban, L., "WaveLAN II: A high-performance wireless LAN for the unlicensed band," Bell Labs Technical Journal, pp. 118-133, Summer 1997.

[8]   Kapoor, R., Chen, L.-J., Lao, L., Gerla, M., Sanadidi, M. Y., "CapProbe: A Simple and Accurate Capacity Estimation Technique,"SIGCOMM 2004.

[9]   Lacage, M., Hossein, M., and Turletti, T. IEEE 802.11 Rate Adaptation: A Practical Approach. In Proceedings of ACM MSWiM 2004.

[10]   Lai, K., Baker, M., *"Measuring Bandwidth,"* IEEE INFOCOM '99

[11]   Li, J., Blake, C., Couto, D., Lee, H. I., and Morris, R., "Capacity of Ad Hoc Wireless Networks," ACM MobiCom 2001.

[12]   Mills, D. L. "Network Time Protocol Specification, Implementation and Analysis," RFC 1305, March 1992.

[13]   Moon, S. B., Skelly, P., and Towsley, D., "Estimating and Removal of Clock Skew from Network Delay Measurements," IEEE Infocom 1999.

[14]   Paxson, V., "On Calibrating Measurements of Packet Transit Times," ACM SIGMETICS 1998.

[15]   Qiao, D., Choi, S. Jain, A., and S, K. G., "MiSer: An Optimal Low-Energy Transmission Strategy for IEEE 802.11a/h", MobiCom 2003.

[16]   Sadeghi, S., Kanodia, V., Sabharwal, A., and Knightly, E., "Opportunistic Media Access for Multirate Ad Hoc Networks," MobiCom 2002.

[17] Zhang, L., Liu, Z., and Xia, C. H., "Clock Synchronization Algorithms for Network Measurements," IEEE Infocom 2002.

[18] OAR, *ww.ece.rice.edu/networks/software/OAR/OAR.html*

[19] NS-2 Simulator, www.mash.cs.berkeley.edu/ns/

Mobility Assessment for MANETs Requiring Persistent Links

Sanlin Xu   Kim Blackmore   Haley Jones

Department of Engineering, Australian National University

Abstract

Mobile ad hoc networks (MANETs) have inherently dynamic topologies. It is important to be able to determine the reliability of the communication paths created under these difficult circumstances. For this purpose, mobility metrics have been proposed in the literature, but most existing research is based on simulation results and empirical analysis. We consider two metrics, link persistence and path persistence, and develop an analytical framework to derive their exact expressions as well as the corresponding link residual time and path residual time, under a random mobility environment. Such exact expressions constitute precise mathematical relationships between the network connectivity and the mobility of mobile nodes. This framework could be used to develop efficient algorithms for medium access control, or to optimize existing network routing protocols.

1 Introduction

Mobile ad hoc networks are comprised of mobile nodes communicating via wireless links. Due to the mobility, communication paths are dynamic, affecting path reliability. Meaningful analysis of the reliability of these constituent links is crucial to understanding the reliability of the paths themselves. Node mobility leads to changes in network topology resulting in link additions and breakages, and alteration of traffic patterns and/or traffic distributions. Routing protocols [1, 2, 3] based on mobility metric prediction have been shown to increase the packet delivery ratio and reduce routing overhead.

In this paper, we consider the notions of *link persistence* and *path persistence* to describe the continuous link and path availabilities, respectively, for an active communication route. The link (path) persistence is the probability that a link (path) lasts constantly until a future time, k, given that it existed at time 0. From theoretical analyses of the link (path) persistence, we derive

the expected residual time for an active link (path). The residual time is evaluated deterministically in [1, 2], by simulation in [4, 5, 6], and by calculation in [7]. Note that the various mobility measures have been given different names by different authors.

A similar measure to link persistence, link availability, is considered by Jiang *et al* [8]. The link availability is approximated as the ratio of the mean time a link will be continuously available to a link prediction time. The calculation method in [8] relies on a parameter, ε, that must be experimentally determined. In [9], link availability is also considered but, in this case, a link is considered to be available at k even if it has undergone failure during the time between 0 and k. Further, in [7] an expression for link availability, using a simple *straight line* mobility model, is derived.

Our notion of link or path persistence requires a random mobility model [10] governing the behaviour of the nodes in the network. We assume nodes move according to a generalisation of the Random Walk Mobility Model. Such an assumption is clearly not realistic, but may be useful as an aid to predicting link reliability for routing purposes [9]. Moreover, random mobility models are regularly used for protocol evaluation [5], so our work is important to facilitate comparison of the evaluation environment with practical implementation environments.

This paper is organised as follows. In Section 2 we present definitions for the mobility metrics investigated in this paper. In Section 3 we develop an expression for the PDF of the separation distance between an arbitrary pair of mobile nodes after one epoch, and generalize the results to a network of nodes with i.i.d. mobility. In Section 4 we develop a Markov chain model for the evolution of the separation distance between two nodes. This Markov model is applied, in Section 5, to the determination of expressions for a range of mobility metrics. In Section 6 we compare our theoretical and simulation results. Finally, we present conclusions in Section 7.

2 Mobility Metric Definitions

While fading links are the norm in wireless communication networks [11], the existing literature on mobility in MANETs concentrates on "binary" links. That is, links are understood to be "on" or "off" at any point in time. Schemes which use network topology information are sensitive to the length of time for which a link is consistently "on". Therefore, we consider the following metrics, which assume that the link is "on", and ask how long it will continue to be "on".

- *Link Persistence, $\mathcal{P}_L(k)$:* Given an active link between two nodes at time *0*, the *persistence* of this link, $\mathcal{P}_L(k)$, until (at least) epoch k is defined as the probability that the link continuously lasts until epoch k.

$$\mathcal{P}_L(k) \triangleq \Pr\{\text{Lasts until } k | \text{Available at } 0\} \quad (1)$$

- *Link Residual Time, $\mathcal{R}_L$:* Given an active link between two nodes at time *0*, the *link residual time*, $\mathcal{R}_L$, is the length of time for which the link will continue to exist until it is broken.

- *Path Persistence, $\mathcal{P}_P(k; h)$:* Given an active path with h hops between two nodes at time *0*, the path persistence, at epoch k, is defined as the probability that the path will continuously last until at least epoch k.

$$\mathcal{P}_P(k; h) \triangleq \Pr\{\text{Last until } k | \text{Available at } 0\} \quad (2)$$

- *Path Residual Time, $\mathcal{R}_P(h)$:* Given an active path with h hops between two nodes at time *0*, the *path residual time*, $\mathcal{R}_P(h)$, is the length of time for which the path will continue to exist.

3 Node Separation After One Epoch

In this paper, we assume that each mobile node moves with a velocity uniformly distributed in both speed, V, and direction, Φ. Both the speed and direction change in each epoch but are constant for the duration of an epoch, and independent of each other. The speed has mean, $\overline{v}$, and variance $\delta^2/3$, such that $V \in \{\overline{v} - \delta, \overline{v} + \delta\}$. The direction, Φ, is uniformly distributed in $[0, 2\pi)$. This random mobility model is widely used to analyze route stability in multi-hop mobile environments [2, 7, 12].

The status of a wireless link depends on numerous system and environmental factors that affect transmitter and receiver transmission range. A widely applied, albeit optimistic, model is used in this paper, whereby transmission range is approximated by a circle of radius r corresponding to a signal strength threshold. If the separation distance between a given pair of nodes is less than
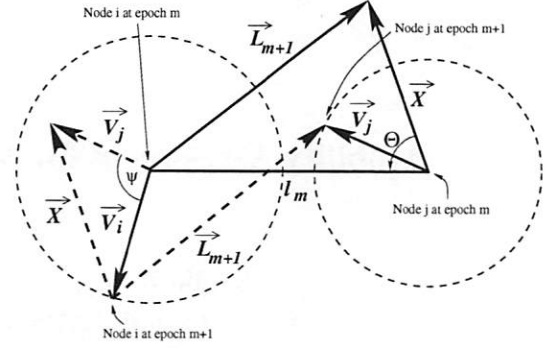

Figure 1: Relationship between the node movement vectors $\overrightarrow{V}_i$ and $\overrightarrow{V}_j$ of nodes i and j, relative movement vector, $\overrightarrow{X}$, initial separation distance, l_m, and separation vector, $\overrightarrow{L}_{m+1}$.

r, it is assumed the link between them is active. Let the random variable representing the separation distance between two nodes at epoch m be L_m, and let l_m denote an instance of L_m. The PDF, $f_{L_{m+1}|L_m}(l_{m+1}|l_m)$, forms a basis for the derivations of expressions for the metrics given in Section 2. We derive this PDF below.

3.1 Relative Movement Between 2 Nodes

To determine $f_{L_{m+1}|L_m}(l_{m+1}|l_m)$, we begin with the PDF of the *relative movement* between a given pair of nodes, labelled i and j. The relationship between the relative movement vector, $\overrightarrow{X}$, in any given epoch, and the node velocity vectors, $\overrightarrow{V}_i$ and $\overrightarrow{V}_j$, is $\overrightarrow{X} = \overrightarrow{V}_j - \overrightarrow{V}_i$, as depicted in Fig. 1. Let X be the random variable representing the magnitude of $\overrightarrow{X}$, similarly for V_i and V_j. Let Ψ be the angle between $\overrightarrow{V}_i$ and $\overrightarrow{V}_j$. As $X = \sqrt{V_i^2 + V_j^2 - 2V_iV_j \cos \Psi}$, we use the Jacobian transform to obtain the joint PDF:

$$f_{X,V_i,V_j}(x, v_i, v_j) = \frac{2x f_{\Psi,V_i,V_j}(\psi, v_i, v_j)}{\sqrt{2v_i^2v_j^2 + 2v_i^2x^2 + 2v_j^2x^2 - v_i^4 - v_j^4 - x^4}}. \quad (3)$$

As Ψ is uniformly distributed in $[0, \pi)$ and V_i, V_j and Ψ are independent, we can simplify (3) to

$$f_{X,V_i,V_j}(x, v_i, v_j) = \frac{x}{2\pi\delta^2 \sqrt{2v_i^2v_j^2 + 2v_i^2x^2 + 2v_j^2x^2 - v_i^4 - v_j^4 - x^4}}. \quad (4)$$

Then the marginal PDF of the magnitude of the relative movement is,

$$f_X(x) = \int_{\overline{v}-\delta}^{\overline{v}+\delta} \int_{\overline{v}-\delta}^{\overline{v}+\delta} f_{X,V_i,V_j}(x, v_i, v_j) dv_i dv_j. \quad (5)$$

Thus, (4) and (5) describe the behaviour of the relative distance, X, between a given pair of nodes, i and j, in any one epoch.

3.2 Calculation of the Separation Distance

From Fig. 1, $\vec{L}_{m+1} = \vec{l}_m + \vec{X}$. Let Θ from Fig. 1 be uniformly distributed in the interval $[0, 2\pi)$. The conditional PDF is

$$f_{L_{m+1}|L_m,\Theta}(l_{m+1}|l_m,\theta) = f_{X,\Theta}(x,\theta) \left| \frac{\partial(x,\theta)}{\partial(l_{m+1},\theta)} \right|$$
$$= f_{X,\Theta}(x,\theta) \frac{l_{m+1}}{\sqrt{l_{m+1}^2 - l_m^2 \sin^2\theta}}, \qquad (6)$$

where $x = l_m \cos\theta \pm \sqrt{l_{m+1}^2 - l_m^2 \sin^2\theta}$. Since the magnitude and phase are independent, we obtain

$$f_{L_{m+1}|L_m}(l_{m+1}|l_m) = \frac{2}{\pi}\int_0^\pi f(x) \frac{l_{m+1}}{\sqrt{l_{m+1}^2 - l_m^2 \sin^2\theta}} d\theta, \qquad (7)$$

where $0 \le x \le \max(V_i + V_j)$, due to the triangular relationship between $\vec{X}$, $\vec{V_i}$ and $\vec{V_j}$.

Next, we use $f_{L_{m+1}|L_m}(l_{m+1}|l_m)$ to derive the node separation distance PDF after k epochs.

4 Markov Chain Model of Separation

In Section 3 we derived the PDF of the distance between a given pair of nodes in epoch $m+1$, $m \in Z^+$, given the separation distance in epoch m. That is, the separation distance in epoch m can be represented as a random variable, L_m. The evolution of the separation distance can be represented by a sequence of random variables, $\{\cdots, L_{m-1}, L_m, L_{m+1}, \cdots\}$. So, the relationship between the separation distances in a sequence of epochs readily lends itself to modelling via a Markov chain process.
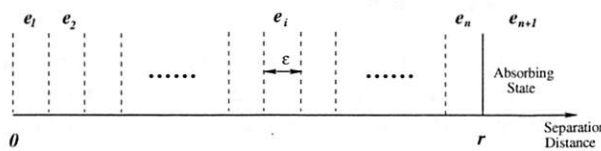
4.1 Transition Model

Figure 2: State space for distance between a pair of nodes, where separations outside of the transmission range (absorbing state) result in a link being discarded.

We first define the state space governing the range of node separation distances, $0 \le l_m < \infty$. We divide the interval $[0, r)$ into n bins of width ε. If a link is active, l_m falls into one of these bins, as shown in Fig. 2. The ith bin corresponds to the ith state, denoted e_i. The $(n+1)$th state, corresponding to $l_m > r$, is defined as the *absorbing state*. A separation distance in the absorbing state indicates a broken link. If the nodes move back within communication range, a new link is considered to have been formed. The distance, L_m, is in e_i if the following conditions hold,

$$\begin{cases} L_m \in e_i \Leftrightarrow \varepsilon(i-1) \le l_m < \varepsilon i & i \in [1,n], \\ L_m \in e_{n+1} \Leftrightarrow l_m > r & i = n+1. \end{cases} \qquad (8)$$

4.2 Initial Condition Vector

The *initial condition vector* denotes the probability of the initial separation distance, L_0, being in each state at time 0. It has entries

$$p_{e_i}(0) = \Pr(L_0 \in e_i). \qquad (9)$$

We assume nodes are initially uniformly distributed. Since they move according to a random walk, they remain (approximately) uniformly distributed in the transmission region. Therefore, if a link exists the initial separation distance, L_0, has a linear distribution [13]:

$$f_{L_0}(l_0) = \frac{2l_0}{r^2}, \quad 0 \le l_0 \le r, \qquad (10)$$

$$p_{e_i}(0) = (2i-1)\frac{\epsilon^2}{r^2}, \quad 0 \le i \le n. \qquad (11)$$
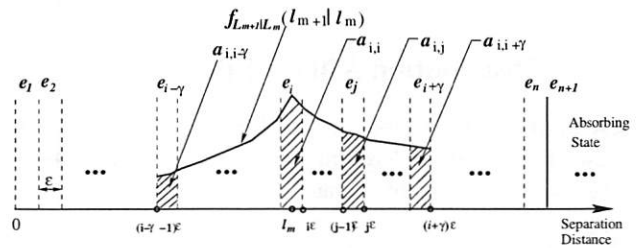
4.3 Transition Matrix

Figure 3: Shows a_{ij}, the probability of transferring from e_i to e_j after one epoch, for various j.

Let L_m be in e_i. Then, after one epoch, L_{m+1}, must be in the range $[l_m - 2(\overline{v}+\delta), l_m + 2(\overline{v}+\delta)]$. This corresponds to L_{m+1} being in e_j such that $j \in [\max(1, i-\gamma), \min(i+\gamma, n+1)]$, where $\gamma = \lceil 2(\overline{v}+\delta)/\varepsilon \rceil$, is the

maximum number of bins that can be traversed in a single epoch. The transition matrix is denoted by

$$
A = \begin{bmatrix} a_{11} & \cdots & a_{1,n} & a_{1,n+1} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n,1} & \cdots & a_{n,n} & a_{n,n+1} \\ 0 & \cdots & 0 & 1 \end{bmatrix}, \quad (12)
$$

where a_{ij} is the probability of transition from e_i to e_j in a given epoch. We note that $\forall\ i,\ j,\ a_{ij} \geq 0$ and $\sum_j a_{ij} = 1$. The last row in (12) represents transition from absorbing state.

To calculate the transition probabilities between non-absorbing states, illustrated in Fig. 3, consider the state transition probabilities at epoch m, and using (7)

$$
a_{ij} = \Pr\left(e_j \leftarrow e_i\right) = \quad (13)
$$
$$
\int_{(j-1)\varepsilon}^{j\varepsilon} \int_{(i-1)\varepsilon}^{i\varepsilon} f_{L_{m+1}|L_m}(l_{m+1}|l_m) f_{L_m}(l_m) dl_m dl_{m+1}.
$$

The PDF $f_{L_m}(l_m)$ changes with m. However, if ε is *sufficiently small*, and e_i is known, we can assume that l_m is approximately uniformly distributed within the ith bin. In this case,

$$
f_{L_m}(l_m) \approx 1/\varepsilon. \quad (14)
$$

Moreover, we can approximate the PDF of the conditioned separation distance from any point in e_i to any point in e_j using the midpoints of the two states:

$$
f_{L_{m+1}|L_m}(l_{m+1}|l_m) \approx f_{L_{m+1}|L_m}\left[(j-1/2)\varepsilon|(i-1/2)\varepsilon\right]. \quad (15)
$$

Thus, from (13), (14) and (15) we can write

$$
a_{ij} \approx \varepsilon f_{L_{m+1}|L_m}\left[(j-1/2)\varepsilon|(i-1/2)\varepsilon\right]. \quad (16)
$$

4.4 Distribution After k Epochs

According to the properties of Markov Chains, we can use A and $P(0)$ to determine the probability of the separation distance being in state e_i after k epochs:

$$
P(k) = [p_{e_1}(k)\ p_{e_2}(k)\ \cdots\ p_{e_{n+1}}(k)] = P(0)\ A^k, \quad (17)
$$

5 Mobility Metric Calculations

Having developed a Markov chain model for the evolution of the node separation, we now derive expressions for the mobility metrics defined in Section 2.

Link Persistence, $\mathcal{P}_L(k)$: The probability of the link being in existence after k epochs is the probability that the separation distance is in a non-absorbing state:

$$
\mathcal{P}_L(k) = \sum_{i=1}^{n} p_{e_i}(k) = 1 - p_{e_{n+1}}(k), \quad (18)
$$

where $p_{e_i}(k)$ can be calculated from (17).

Link Residual Time, $\mathcal{R}_L$: The CDF, $F_{\mathcal{R}_L}(k)$, of the link residual time, is:

$$
F_{\mathcal{R}_L}(k) = \Pr\{\mathcal{R}_L \leq k\} = 1 - \mathcal{P}_L(k). \quad (19)
$$

Then, the PDF of the link residual time is

$$
f_{\mathcal{R}_L}(k) = \Pr\{\mathcal{R}_L = k\} = \mathcal{P}_L(k-1) - \mathcal{P}_L(k). \quad (20)
$$

The expected value of the link residual time is

$$
E\{\mathcal{R}_L\} = \sum_{k=1}^{\infty} k\, f_{\mathcal{R}_L}(k) = \sum_{k=1}^{\infty} k\, [\mathcal{P}_L(k-1) - \mathcal{P}_L(k)]. \quad (21)
$$

Path Persistence, $\mathcal{P}_P(k; h)$: For a path to persist, each of the component links must persist:

$$
\mathcal{P}_P(k; h) = \prod_{i=1}^{h} \mathcal{P}_L(k). \quad (22)
$$

Path Residual Time, $\mathcal{R}_P(h)$: The CDF, $F_{\mathcal{R}_P(h)}(k; h)$, of the path residual time is

$$
F_{\mathcal{R}_P(h)}(k; h) = \Pr\{\mathcal{R}_P(h) \leq k\} = 1 - \mathcal{P}_P(k; h) \quad (23)
$$

Therefore, the PDF of the path residual time is

$$
f_{\mathcal{R}_P(h)}(k; h) = \mathcal{P}_P(k-1; h) - \mathcal{P}_P(k; h). \quad (24)
$$

And, the expected value of the path residual time is

$$
E\{\mathcal{R}_P(h)\} = \sum_{k=1}^{\infty} k\, [\mathcal{P}_P(k-1; h) - \mathcal{P}_P(k; h)]. \quad (25)
$$

6 Simulation Results

We verify all analytically derived expressions for the mobility metrics by simulation. The simulations were conducted with MNs moving according to the description in Section 3.

The network consisted of 100 nodes initially placed randomly in a square plane of side 1000 distance units. We use the generic term "units" rather than, say, m or km because it is the relative and not the absolute distances

that are important. Each MN had a maximum transmission range of $r = 100$ units, with between 1000 and 5000 epochs per trial for 3000 trials.

Figure 4 illustrates the comparisons of the simulation results to the theoretical calculations. The link persistence, shown in Fig. 4(a), decreases with increasing simulation time and, at a greater rate with increasing ratio of mean node speed to transmission range, $\overline{v}/r$. Further, the path persistence drops off at a greater rate than the link persistence, for the same mean node speed, as would be expected. The path persistence also drops off more quickly with an increased number of hops, as there is more chance of an individual link breaking.

In the bounded simulation environment, MNs were "reflected" back into the simulation area, if their movement would otherwise take them outside. Consequently, nodes near the edge are more likely to remain in transmission range, and the link persistence is artificially increased, compared to that for the unbounded simulation area. The experimental results for the bounded area are still close to the calculated results, as expected, though not as well matched.

The expected link and path residual times have been plotted against the second order of $r/\overline{v}$, each showing a linear relationship, particularly for larger ratios. As expected, $E\{\mathcal{R}_P\}$ is much lower than $E\{\mathcal{R}_L\}$ for the same communication range to speed ratio. Finally, the probability distributions of $\mathcal{R}_L$ and $\mathcal{R}_P$ show that the path residual time is more likely to have a shorter length, in epochs, than the link residual time.

7 Conclusions

Frequent changes in network topology caused by mobility in MANETs imposes great challenges for developing efficient routing algorithms. The theoretical analysis framework presented in this paper provides a better understanding of network behavior under mobility and some fundamental work on the issue of path stability. We propose link persistence and path persistence for evaluating link and path stability. The Markov chain model used in this paper, has enabled us to accurately determine a series of mobility metrics. These calculations are useful for comparison of artificial mobility behaviours with actual network implementation scenarios. The analytical results can be readily applied to various adaptive routing protocols that use corresponding mobility metrics.

References

[1] W. Su, S. Lee, and M. Gerla, "Mobility Prediction and Routing in Ad Hoc Wireless Networks," *Int. J. of Network Management*, 2000.

[2] P. Samar and S. B. Wicker, "On the Behavior of Communication Links of a Node in a Multi-Hop Mobile Envirionment," in *Proceedings of Mobi-Hoc*, May 2004, pp. 145–156.

[3] L. Qin and T. Kunz, "Increasing Packet Delivery Ratio in DSR by Link Prediction," in *Proc. 36th Hawaii Int. Conf. on Syst. Sciences*, Jan. 2002.

[4] M. Gerharz, C. Waal, and P. Martini, "Strategies for Finding Stable Paths in Mobile Wireless Ad Hoc Networks," in *Proc. of IEEE Conf. on Local Computer Networks (LCN)*, Oct. 2003, pp. 130–139.

[5] F. Bai, N. Sadagopan, and A. Helmy, "IMPORTANT a Framework to Systematically Analyze the Impact of Mobility on Performance of Routing Protocols for Ad Hoc Networks," in *IEEE INFOCOM*, vol. 2, Apr. 2003, pp. 825–835.

[6] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy, "PATHS: Analysis of PATH Duration Statistics and Their Impact on Reactive MANET Routing Protocols," in *MobiHoc 03*, June 2003, pp. 246–256.

[7] D. Yu and H. Li, "Path Availability in Ad Hoc Networks," *IEEE ICT Conf.*, vol. 1, pp. 383–387, 2003.

[8] S. Jiang, "An Enhanced Prediction-based Link Availability Estimation for Manets," *IEEE Trans. Commun.*, vol. 52, no. 2, pp. 183–186, Feb. 2004.

[9] A. B. McDonald and T. Znati, "A Mobility Based Framework for Adaptive Clustering in Wireless Ad-Hoc Networks," *IEEE J. Select. Areas Commun.*, vol. 17, no. 8, pp. 1466–1487, Aug. 1999.

[10] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Commun. and Mobile Computing (WCMC)*, vol. 5, no. 2, pp. 483–502, 2002.

[11] H. Jones, S. Xu, and K. Blackmore, "Link Ratio for Ad Hoc Networks in a Rayleigh Fading Channel," *WITSP, Australia*, Dec. 2004.

[12] B. An and S. Papavassiliou, "An Entropy-Based Model for Supporting and Evaluating Route Stability in Mobile Ad Hoc Wireless Networks," *IEEE Commun. Lett.*, vol. 6, no. 8, pp. 328–330, 2002.

[13] D. Hong and S. Rappaport, "Traffic Model and Performance Analysis for Cellular Mobile Radio Telephone Systmes with Prioritized and Nonprioritized Handoff Procedures," *IEEE Trans. Veh. Technol.*, vol. 35, pp. 77–92, Aug. 1986.

THE USENIX ASSOCIATION

Since 1975, the USENIX Association has brought together the community of developers, programmers, system administrators, and architects working on the cutting edge of the computing world. USENIX conferences have become the essential meeting grounds for the presentation and discussion of the most advanced information on new developments in all aspects of advanced computing systems. USENIX and its members are dedicated to:

- problem-solving with a practical bias
- fostering innovation and research that works
- communicating rapidly the results of both research and innovation
- providing a neutral forum for the exercise of critical thought and the airing of technical issues

For a complete listing of member benefits, see http://www.usenix.org/membership/bens.html.

Want more information about USENIX? See http://www.usenix.org/ or contact:
USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710 USA
Phone: 510-528-8649 Fax: 510-548-5738 Email: *office@usenix.org*

Thanks to USENIX Supporting Members

❖ Addison-Wesley/Prentice Hall PTR ❖ Ajava Systems, Inc. ❖ AMD ❖ Asian Development Bank ❖

❖ Atos Origin BV ❖ Cambridge Computer Services, Inc. ❖ Delmar Learning ❖

❖ DoCoMo Communications Laboratories USA, Inc. ❖ Electronic Frontier Foundation ❖

❖ Hewlett-Packard ❖ IBM ❖ Intel ❖ Interhack MacConnection ❖ The Measurement Factory ❖

❖ Microsoft Research ❖ ❖ Oracle ❖ OSDL ❖ Perfect Order ❖ Portlock Software ❖ Raytheon ❖

❖ Sun Microsystems, Inc. ❖ Taos ❖ Tellme Networks ❖ UUNET Technologies, Inc. ❖ Veritas Software ❖

ACM SIGMOBILE

ACM SIGMOBILE is the Association for Computing Machinery's Special Interest Group on Mobility of Systems, Users, Data, and Computing. Founded in 1947, ACM is the world's first educational and scientific computing society. Today, ACM serves a membership of more than 80,000 computing professionals in more than 100 countries in all areas of industry, academia, and government.

SIGMOBILE is the primary international organization dedicated to addressing the latest developments in all areas of mobile computing and wireless and mobile networking. SIGMOBILE has members from around the world, from academic organizations, industry research and development, government, and other interested individuals. Members of SIGMOBILE are active in the development of new technologies and techniques for mobile and wireless computing and communications.

As a member of SIGMOBILE, you will receiver our quarterly journal, *Mobile Computing and Communications Review* (MC R). You will also be able to receive announcements via our moderated members-only email distribution list, keeping you informed of the latest happenings in our field, including new opportunities to share your research and to meet with friends and colleagues at conferences and other events. SIGMOBILE members also are eligible for the lowest generally available registration fee for any event sponsored or co-sponsored by SIGMOBILE, and for the many events sponsored by other organizations offered in-cooperation with SIGMOBILE.

For more information about ACM SIGMOBILE, visit us on the web at *http://www.sigmobile.org/*. You may also contact the ACM Membership Services Department by email at *acmhelp@acm.org* or by telephone at 1-800-342-6626 (U.S. and Canada) or +1-212-626-0500 (outside U.S.).